

AUTOMATED CODE COMPLIANCE CHECKING IN THE  
CONSTRUCTION DOMAIN USING SEMANTIC NATURAL LANGUAGE  
PROCESSING AND LOGIC-BASED REASONING

BY

JIANSONG ZHANG

DISSERTATION

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Civil Engineering  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Doctoral Committee:

Assistant Professor Nora El-Gohary, Chair  
Professor Khaled El-Rayes  
Associate Professor Liang Liu  
Associate Professor Corina Roxana Girju  
Assistant Professor Mani Golparvar-Fard

## ABSTRACT

Construction projects must comply with various regulations. The manual process of checking the compliance with regulations is costly, time consuming, and error prone. With the advancement in computing technology, there have been many research efforts in automating the compliance checking process, and many software development efforts led by industry bodies/associations, software companies, and/or government organizations to develop automated compliance checking (ACC) systems. However, two main gaps in the existing ACC efforts are: (1) manual effort is needed for extracting requirements from regulatory documents and encoding these requirements in a computer-processable rule format; and (2) there is a lack of a semantic representation for supporting automated compliance reasoning that is non-proprietary, non-hidden, and user-understandable and testable. To address these gaps, this dissertation proposes a new ACC method that: (1) utilizes semantic natural language processing (NLP) techniques to automatically extract regulatory information from building codes and design information from building information models (BIMs); and (2) utilizes a semantic logic-based representation to represent and reason about the extracted regulatory information and design information for compliance checking. The proposed method is composed of four main methods/algorithms that are combined in one computational framework: (1) a semantic, rule-based method and algorithm that leverage NLP techniques to automatically extract regulatory information from building codes and

represent the extracted information into semantic tuples, (2) a semantic, rule-based method and algorithm that leverage NLP techniques to automatically transform the extracted regulatory information into logic rules to prepare for automated reasoning, (3) a semantic, rule-based information extraction and information transformation method and algorithm to automatically extract design information from BIMs and transform the extracted information into logic facts to prepare for automated reasoning, and (4) a logic-based information representation and compliance reasoning schema to represent regulatory and design information for enabling the automated compliance reasoning process. To test the proposed method, a building information model test case was developed based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences. The test case was checked for compliance with a randomly selected chapter, Chapter 19, of the International Building Code 2009. Comparing to a manually developed gold standard, 87.6% precision and 98.7% recall in noncompliance detection were achieved, on the testing data.

## **ACKNOWLEDGEMENTS**

First and foremost, I would like to express my profound gratitude to my advisor Professor Nora El-Gohary for her guidance and support. It has been an honor to be her first Ph.D. student. I appreciate all her contributions of time and support to make my Ph.D. experience productive and stimulating. I also want to thank my committee members Professor Khaled El-Rayes, Professor Liang Liu, Professor Corina Roxana Girju, and Professor Mani Golparvar-Fard for their advice, time, comments, and encouragement.

I would like to give special thanks to my love, Cheng Lu, for being incredibly supportive and patient during the course of this Ph.D. program. Special thanks also go to my parents and my big family for all their love, trust, encouragement, and unconditional support.

The members of the Newmark Civil Engineering Laboratory, especially the members of our research lab, have contributed immensely to both my professional and personal life at the University of Illinois at Urbana-Champaign. They have been a source of help, friendship, communication, and spiritual support. My thanks also go to my friends from many other departments at the University of Illinois at Urbana-Champaign, represented by many members of Chinese Students and Scholars Association.

I gratefully acknowledge the National Science Foundation and the University of Illinois at Urbana-Champaign Campus Research Board who generously supported/sponsored my research.

## TABLE OF CONTENTS

1	CHAPTER 1 – INTRODUCTION .....	1
1.1	Motivation and Overview .....	1
1.2	State of the Art in Automated Compliance Checking and Practical Gaps .....	3
1.3	Proposed Approach.....	6
1.4	Knowledge Gaps.....	10
1.5	Problem Statement.....	12
1.6	Research Objectives and Questions .....	12
1.7	Research Methodology and Tasks .....	16
1.8	Contribution and Significance .....	27
2	CHAPTER 2 – LITERATURE REVIEW .....	33
2.1	Automated Compliance Checking (ACC) in the Construction Domain .....	33
2.2	Natural Language Processing (NLP) .....	42
2.3	Information Extraction from Building Information Models .....	54
2.4	Automated Reasoning.....	62
2.5	Semantic Modeling and Information Processing .....	66
2.6	Machine Learning Algorithms .....	76

3	CHAPTER 3 – AUTOMATED INFORMATION EXTRACTION FROM BUILDING CODES .....	80
3.1	Comparison to the State of the Art .....	80
3.2	Proposed Information Extraction Method and Algorithm .....	83
3.3	Experimental Testing and Evaluation .....	102
4	CHAPTER 4 – AUTOMATED INFORMATION TRANSFORMATION OF REGULATORY INFORMATION .....	118
4.1	Comparison to the State of the Art .....	118
4.2	Proposed Information Transformation Method and Algorithm .....	119
4.3	Experimental Testing and Evaluation .....	134
5	CHAPTER 5 – SEMIAUTOMATED IFC EXTENSION .....	154
5.1	Comparison to the State of the Art .....	154
5.2	Proposed IFC Extension Method and Algorithm.....	157
5.3	Experimental Testing and Evaluation .....	176
6	CHAPTER 6 – AUTOMATED INFORMATION EXTRACTION FROM BUILDING INFORMATION MODELS AND TRANSFORMATION OF DESIGN INFORMATION	200
6.1	Comparison to the State of the Art .....	200
6.2	Proposed BIM Information Extraction and Transformation Method and Algorithm .....	203
6.3	Experimental Testing and Evaluation .....	219

7	CHAPTER 7 – LOGIC-BASED INFORMATION REPRESENTATION AND COMPLIANCE REASONING SCHEMA.....	225
7.1	Comparison to the State of the Art .....	225
7.2	Proposed Information Representation and Compliance Reasoning Schema .....	229
7.3	Software Implementation.....	244
7.4	Experimental Testing and Evaluation.....	251
8	CHAPTER 8 – PROTOTYPE SYSTEM IMPLEMENTATION AND EXPERIMENTAL TESTING .....	260
8.1	System Architecture.....	260
8.2	System Implementation .....	262
8.3	Data for System Testing .....	269
8.4	Evaluation Metrics.....	270
8.5	Results and Discussion .....	271
9	CHAPTER 9 – CONCLUSIONS, CONTRIBUTIONS, LIMITATIONS, AND RECOMMENDATIONS FOR FUTURE RESEARCH.....	275
9.1	Conclusions .....	275
9.2	Contributions to the Body of Knowledge .....	282
9.3	Limitations and Recommendations for Future Research .....	289
10	REFERENCES .....	295

11	APPENDIX A: ANNOTATION GUIDELINES FOR REGULATORY INFORMATION EXTRACTION .....	320
12	APPENDIX B: ANNOTATION GUIDELINES FOR REGULATORY INFORMATION TRANSFORMATION.....	323



# 1 CHAPTER 1 – INTRODUCTION

## 1.1 Motivation and Overview

Construction projects are governed by a multitude of federal, state, and local regulations, such as the International Building Code (IBC), the ADA Standards for Accessible Design, the International Fire Code, the International Energy Conservation Code, the OSHA’s Cranes and Derricks in Construction, the Illinois Accessibility Code, the Illinois Energy Conservation Code, the Illinois Plumbing Code, and the Municipal Code of Chicago. Each regulation has a large set of provisions. For example, the IBC 2006 is composed of 329 sections, where each section includes several to tens of provisions that address a variety of requirements (e.g., safety, environmental).

Due to the large number of construction regulatory documents, the variability of their provisions in terms of formatting and semantics, and the large amount and complexity of the information they describe; like other manual processes (Boken and Callaghan 2009), the manual process of regulatory compliance checking is time-consuming, costly, and error-prone. For example, in the city of Mesa, Arizona, the turn-around time for a commercial building plan review is 18 business days, with a fee assessed at a rate of \$90 per hour (City of Mesa 2012). Failure to comply with regulations could further result in incurring much higher costs. For example, Wal-Mart Stores Inc. was fined \$1 million due to violation of storm-water regulations (US EPA 2004; Salama and El-Gohary 2011). Automated

compliance checking (ACC) is expected to reduce the cost, time, and errors of compliance checking (Tan et al. 2010; Eastman et al. 2009).

With the advancement in computing technology, there have been many research efforts in automating the compliance checking process (e.g., Garrett and Fenves 1987; Delis and Delis 1995; Han et al. 1997; Lau and Law 2004; Eastman et al. 2009; Tan et al. 2010). Larger research and software development efforts for automated building code checking led by industry bodies/associations, software companies, and/or government organizations include Solibri Model Checker (Corke 2013), EPLAN/BIM led by Fiatech (Fiatech 2011), Construction and Real Estate NETwork (CORENET) led by the Singapore Ministry of National Development (Singapore Building and Construction Authority 2006), REScheck and COMcheck led by the U.S. Department of Energy (US DOE 2011), SMARTcodes led by the International Code Council (ICC 2011), and Avolve Software (Avolve Software Corporation 2011). Previous research and software development efforts have undoubtedly paved the way for ACC in the in the architectural, engineering, and construction (AEC) industry. However, these efforts are limited in their automation and reasoning capabilities (Zhong et al. 2012); existing ACC systems (1) require manual effort for extracting requirements from textual regulatory documents (e.g., codes) and encoding these requirements in a computer-processable rule format. Rules are either hard-coded into the developed systems or hand-coded as a rule database or set of files. For example, in the most recent effort of the AutoCodes project led by Fiatech (Fiatech 2015), the creation of

regulatory rules requires manual extraction and encoding effort; and (2) lack a semantic representation for supporting automated compliance reasoning that is non-proprietary, non-hidden, and user-understandable and testable.

To address these gaps, this dissertation aims at developing a semantic, Natural Language Processing (NLP)-enabled, and logic-enabled system for ACC in the construction domain. NLP is a field utilizing artificial intelligence to enable computers to understand and process natural language text in a human-like manner (Cherpa 1992). Formally-defined logic provides the theoretical basis and utilities for inference-making. Semantic modeling aims at providing the level of knowledge representation that is needed to facilitate such deep levels of information processing and compliance reasoning; it will help in processing applicable regulations and checking compliance of designs to the provisions/rules that are prescribed by those regulations. Semantic NLP techniques will facilitate textual document analysis and processing for the extraction of regulatory information from regulatory documents. Semantic building information modeling-based methods will facilitate the extraction of design information from building information modeling-based designs. Semantic logic-based reasoning techniques will facilitate automated compliance reasoning and analysis.

## **1.2 State of the Art in Automated Compliance Checking and Practical Gaps**

Since the first attempt to computerize building regulations in 1960s (Fenves et al. 1969), efforts for ACC in the construction domain have been ongoing, such as the checking of safety

and reliability requirements of structures (Garrett and Fenves 1987), fire code compliance (Delis and Delis 1995), facility accessibility code compliance (Han et al. 1997), accessibility (Lau and Law 2004), egress, environmental protection, and energy conservation (FIC 2007), construction inspection and quality control (Boukamp and Akinci 2007), building envelope performance (Tan et al. 2007; Tan et al. 2010), and building design (Eastman et al. 2009). In industry, several software systems have been developed for ACC purposes, such as REScheck and COMcheck (US DOE 2011), Avolve Software (Avolve Software Corporation 2011), and Solibri Model Checker (Corke 2013). However, the state-of-the-art development in ACC still requires manual extraction of regulatory provisions/requirements from textual regulatory documents and encoding of these extracted provisions/requirements into a computer-processable rule format. Rules are either hard-coded into the systems or hand-coded as a rule database or set of files. For example, Solibri Model Checker (Corke 2013) currently includes a set of 300 proforma-based rules that allow for some degree of user customization of rules. However, such customization does not allow for the creation of new rules. The development of new rules in Solibri Model Checker requires professional software engineering expertise and deep understanding of the software's environment and data structure (Corke 2013). The software tools developed by OptaSoft for ACC with International Code Council (ICC) codes need major manual data entry and navigation (OptaSoft 2014).

Also, the state-of-the-art development in ACC is limited in terms of offering a semantic

representation that is non-proprietary, non-hidden, and user-understandable and testable. Existing systems typically hard code the specific compliance checking method (for the targeted checking topic) into programs that are procedural and rigid, which makes it difficult or impossible to separate the rules from specific programs to conduct more complex and flexible analysis in another software program. And, they typically utilized proprietary (thus hidden) information representation and reasoning mechanisms, which makes it difficult for the rules to be understood and tested by regulation writers and compliance checking users (Garrett and Palmer 2014). For example, the REScheck system for checking the compliance of residential buildings with energy codes utilizes a checklist of building elements (which have to be manually-created/selected from a pool of existing building components models) to calculate the U-factor  $\times$  Area (UA) for each building assembly to determine the UA for the building overall. It then compares this UA with UA resulted from a building conforming to the code requirements to determine the compliance result of this building. This procedure is strictly procedural and rigid (US DOE 2013). And the rules are hidden from the users, and thus cannot be easily understood or tested by the users. The utilization of proforma-based rules in Solibri Model Checker makes the checking procedure more flexible than the checking procedure in REScheck. However, as mentioned earlier, the proforma-based rules set is fixed, although some rule parameter adjustment is allowed. This renders the checking mechanism in Solibri Model Checker to be also rigid and procedural, although more flexible than the strictly hard-coded procedure such as in the REScheck system, and thus to be limited

in its reasoning capability (Corke 2013). In addition, the proforma-based rules are in a proprietary format, and thus cannot be easily understood and tested by regulation writers and compliance checking users. The creation of totally new rules in Solibri Model Checker must be conducted by Solibri engineers.

### **1.3 Proposed Approach**

#### **1.3.1 Natural Language Processing (NLP) Approach**

In this dissertation, NLP techniques are used to facilitate textual document analysis and processing for the extraction of regulatory information from regulatory documents. In the author's analysis, in comparison to general non-technical text (e.g., news articles, general websites), domain-specific regulatory text is more suitable for automated NLP (i.e., would allow for better interpretability and less ambiguity in automated processing) due to three main text characteristics. First, construction text is likely to have less homonym conflicts than non-technical text. For example, in news articles, the term "bridge" could refer to a structural bridge, the card game, a bridge of understanding, a dental bridge, etc. Second, it is easier to develop an ontology that captures domain knowledge as opposed to an ontology that captures general knowledge (or a wide variety of domains). A domain ontology may enhance automated interpretability and understandability of domain-specific text. Third, construction text is likely to exhibit less co-reference resolution problems. For example, construction regulatory text tends to mention the subjects (e.g., door) for each provision explicitly rather

than referring to the subjects using pronouns (e.g., “it”).

### **1.3.2 Logic-Based Reasoning Approach**

In this dissertation, logic-based reasoning is used to facilitate automated compliance reasoning about regulatory rules and design information. A logic-based representation and reasoning schema for ACC is developed to allow for leveraging the inference-making capabilities of formally-defined logic. A formally-defined logic could represent and reason about the complicated logic relations in construction regulations more efficiently than procedural programming languages like C programming language. Logic has been essential in many automated reasoning systems (Portoraro 2011). Different types of formally-defined logic with varying degrees of descriptive and reasoning capabilities have been developed to support automated reasoning in various domains, such as in robotics (e.g., forming plans for autonomous robots) and artificial intelligence (e.g., automated question answering). The use of formally-defined logic for automated reasoning has successfully solved many famous problems some of which baffled human experts for decades, such as the Robbins Problem in algebra (Robinson and Voronkov 2001; Bundy 2013).

### **1.3.3 Semantic Modeling Approach**

In this dissertation, semantic modeling is used to support content-based information processing and compliance reasoning. A domain-specific ontology is developed and used to (1) conduct NLP in a semantic way and (2) support the logic-based representation and

reasoning schema. An ontology models domain knowledge in form of concept hierarchies, relationships (between concepts), and axioms (El-Gohary and El-Diraby 2010). The domain knowledge captured in an ontology is expected to facilitate the processing of both regulatory text and design information and to guide the needed complex reasoning for compliance checking. For example, ontology-based semantic information extraction (i.e., using meaning/context-related features, in addition to syntax/grammar-related features) is expected to achieve higher performance in comparison to syntactic information extraction (i.e., information extraction using syntactic features only), because domain knowledge (represented in an ontology) could help to identify or distinguish domain-specific terms and meanings (Soysal et al. 2010).

#### **1.3.4 Proposed Framework**

The proposed ACC framework includes a number of elements (as per Figure 1.1): (1) a building code in textual format such as the International Building Code, (2) a building information model (BIM) based on industry foundation classes (IFC) (i.e., .ifc file) to represent building design information, (3) an ontology representing knowledge in the construction domain to support the processing of regulatory and design information and the automated compliance reasoning process, (4) a semantic, rule-based algorithm that leverages NLP techniques to automatically extract regulatory information from building codes and represent the extracted information into semantic tuples, (5) a semantic, rule-based algorithm that leverages NLP techniques to automatically transform the extracted regulatory



information into logic rules to prepare for automated reasoning, (6) a semantic, rule-based information extraction (IE) and information transformation (ITr) algorithm to automatically extract design information from building information models (BIMs) (.ifc file) and transform the extracted information into logic facts to prepare for automated reasoning, (7) a logic-based information representation and compliance reasoning schema to represent regulatory and design information for enabling the automated compliance reasoning process, (8) automated regulatory information extraction from building codes and representation of the extracted information into semantic tuples, (9) automated regulatory information transformation into logic rules, (10) automated design information extraction from BIMs and transformation of the extracted information into logic facts, (11) compliance testing: using logic-based reasoning to check compliance of design information (in the form of logic facts) with regulatory information (in the form of logic rules), and (12) compliance reporting: reporting the results of compliance testing in terms of compliance or noncompliance, including the associated analysis of a noncompliance (e.g., reason of violation). All the above-mentioned processes (element #8 through #12) are facilitated by the domain knowledge captured by the ontology. Elements #8 and #9, additionally, require the use of NLP techniques, in the form of information extraction and information transformation algorithms. Elements #11 and #12 require the use of logic reasoning based on the logic-based information representation and compliance reasoning schema. As such, the framework involves developing a set of methods/algorithms, along with an information representation

and reasoning schema, and combining them into one computational platform: (1) a semantic, rule-based NLP regulatory information extraction (IE) algorithm, (2) a semantic, rule-based NLP regulatory information transformation (ITr) algorithm, (3) a semantic, rule-based design information extraction (IE) and design information transformation (ITr) algorithm, and (4) a logic-based information representation and compliance reasoning schema.

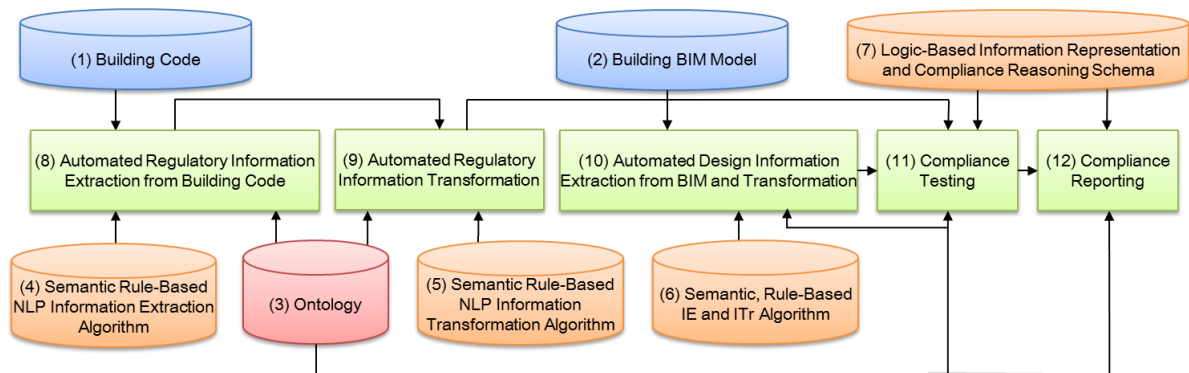


Figure 1.1 Proposed Framework

## 1.4 Knowledge Gaps

### 1.4.1 Gaps in Existing NLP Efforts

There is a lack of methods for automated deep information extraction (IE) from textual sources for supporting compliance checking purposes. The state of the art in NLP has achieved reasonable performances for shallow NLP tasks, whereas it is still being challenged by deep NLP tasks. Shallow NLP conducts partial analysis of a sentence or analyzes a sentence from a specific angle of view [e.g., part-of-speech (POS) tagging, text chunking]. Deep NLP, on the other hand, aims at full sentence analysis with a more complex

understanding of the text toward capturing the entire meaning of sentences (Zouaq 2011). Correspondingly, shallow information extraction extracts specific type(s) of information from a sentence, whereas deep information extraction aims to extract all information expressed by a sentence based on the full analysis of the sentence. Deep NLP/information extraction, thus, requires elaborate knowledge representation and reasoning, which remains to be a challenge for AI (Tierney 2012). For the purpose of ACC, a successful information extraction does require correct understanding of the text source (i.e., textual regulatory documents). This need of a deep level of NLP makes the problem of automated information extraction for compliance checking purposes challenging.

#### **1.4.2 Gaps in Existing Representation Schema for Automated Compliance Reasoning**

There is a lack of a semantic representation schema for construction regulations that is computer-processable, inference engine-independent, and user-understandable and testable to support automated reasoning for ACC (Garrett and Palmer 2014). In an automated reasoning system, the representation schema and reasoning mechanism influence each other. Reasoning needs affect the requirements and structure of the representation and successful reasoning depends on appropriate representations. Finding the right representation is, thus, a key to successful reasoning (Bundy 2013). Building regulations represent requirements using a variety of limits and relationships, and the building regulations are used for checking by different types of domain experts using different softwares. Thus, to support ACC, the computer-processable representation of building regulations needs to be (1) independent from

specific softwares or inference engines, and (2) user-understandable and testable (Garrett and Palmer 2014). Existing ACC systems do not provide such a representation for building regulations that is needed to allow the compliance checking of designs across different softwares and by experts with different knowledge and skills. Such a flexible representation to support compliance checking “remained always a challenge for Artificial Intelligence (AI) experts” (Santos and Farinha 2005). Besides, such a representation schema for construction regulations also requires: (a) a holistic representation to cover design information as well, and (b) a formal, computer-interpretable representation to facilitate automated reasoning.

## **1.5 Problem Statement**

Compliance checking is a costly ‘bottleneck’ in the project delivery process, because it is a highly manual process. Manual code compliance checking is time-consuming, costly, and error-prone. Automating the compliance checking process is expected to reduce time, cost, and error of the process. However, previous efforts towards ACC have been limited, because they (1) require manual effort for extracting requirements from textual regulatory documents (codes) and encoding these requirements in a computer-processable rule format, and (2) lack a semantic representation that is non-proprietary, non-hidden, and user-understandable and testable.

## **1.6 Research Objectives and Questions**

The overall objective of this research is to develop a semantic, natural language processing

(NLP)-enabled, and logic-enabled system (a proof-of-concept prototype) for ACC of BIM-based building designs (.ifc format) with building codes. For building codes, the scope of this dissertation is limited to information represented in natural language sentences, excluding information represented in figures or tables. For BIM-based building designs, the scope of this dissertation is limited to design information in the planning and design phases of a building project [i.e., a level of development (LOD) of 400]. The developed proof-of-concept prototype system will be initially tested in checking the compliance of one BIM test case with one chapter in the International Building Code. Accordingly, six specific objectives are defined:

Objective 1: Develop a semantic, rule-based NLP algorithm for automatically extracting regulatory information from textual regulatory documents (i.e., building codes) for supporting ACC in the construction domain.

Research Questions: How to automatically extract information from textual regulatory documents with a sufficient performance (in terms of precision and recall) for compliance reasoning purposes? What are the necessary features to represent domain-specific text for information extraction (IE)? Would the use of a semantic information extraction approach result in the desired performance in this application? Would the use of a rule-based information extraction approach result in the desired performance in this application?

Objective 2: Develop a semantic, rule-based algorithm for automatically transforming the extracted regulatory information into logic rules for supporting ACC in the construction domain.

Research Questions: How to automatically transform the extracted regulatory information into a logic representation (consisting of logic rules) that would be ready for automated reasoning? How to map the extracted regulatory information to the logic rules? This mapping is expected to: (a) be more complicated than a simple one-to-one mapping; (b) be dependent on the semantic meaning of the extracted information (i.e., the concept or relation each extracted information instance is associated with); and (c) contain possible conflicts in the extracted information.

Objective 3: Develop a semantic, rule-based and machine learning-based algorithm for semiautomatically extending the industry foundation classes (IFC) schema to facilitate the representation of ACC-related information in building information models (BIMs).

Research Questions: How to automatically extract regulatory concepts from building codes with a sufficient performance (in terms of precision and recall) for representing ACC-related information? How to automatically find the most related IFC concept for each regulatory concept? How to automatically find the relationship between regulatory concepts and their most related IFC concepts? How to automatically integrate the regulatory concepts into the IFC schema? How to integrate user judgements into the IFC

extension process to eliminate/reduce possible errors of the automated processes?

Objective 4: Develop a semantic, rule-based algorithm for automatically extracting design information from building information models (BIMs) and automatically transforming the extracted design information into logic facts for supporting ACC in the construction domain.

Research Questions: How to automatically extract information from BIMs with a sufficient performance (in terms of precision and recall) for compliance reasoning purposes? How to automatically transform the extracted information into a logic-based representation (consisting of logic facts) that would be ready for automated reasoning? How to handle missing information or information not available in BIMs?

Objective 5: Develop a logic-based information representation and compliance reasoning schema for representing, both, regulatory information and design information to prepare for utilizing the inference-making capabilities of logic reasoners for supporting ACC in the construction domain.

Research Questions: How to represent the types of regulatory information and design information into a formal information representation schema that could be used to guide the information processing and automated compliance reasoning processes? This representation schema is required to be generalized and flexible to allow for the deep representation of all information (i.e., all concepts and relations) in a regulatory provision regardless of the type, length, and complexity of the provision (sentence).

Objective 6: Integrate and implement all developed methods and algorithms in one platform: a proof-of-concept prototype system that conducts ACC based on regulatory text and a BIM (.ifc file).

Research Questions: How to integrate all developed methods and algorithms in one platform with a sufficient performance (in terms of precision and recall) in noncompliance detection?

## **1.7 Research Methodology and Tasks**

The methodology is composed of seven main tasks, as illustrated in Figure 1.2.



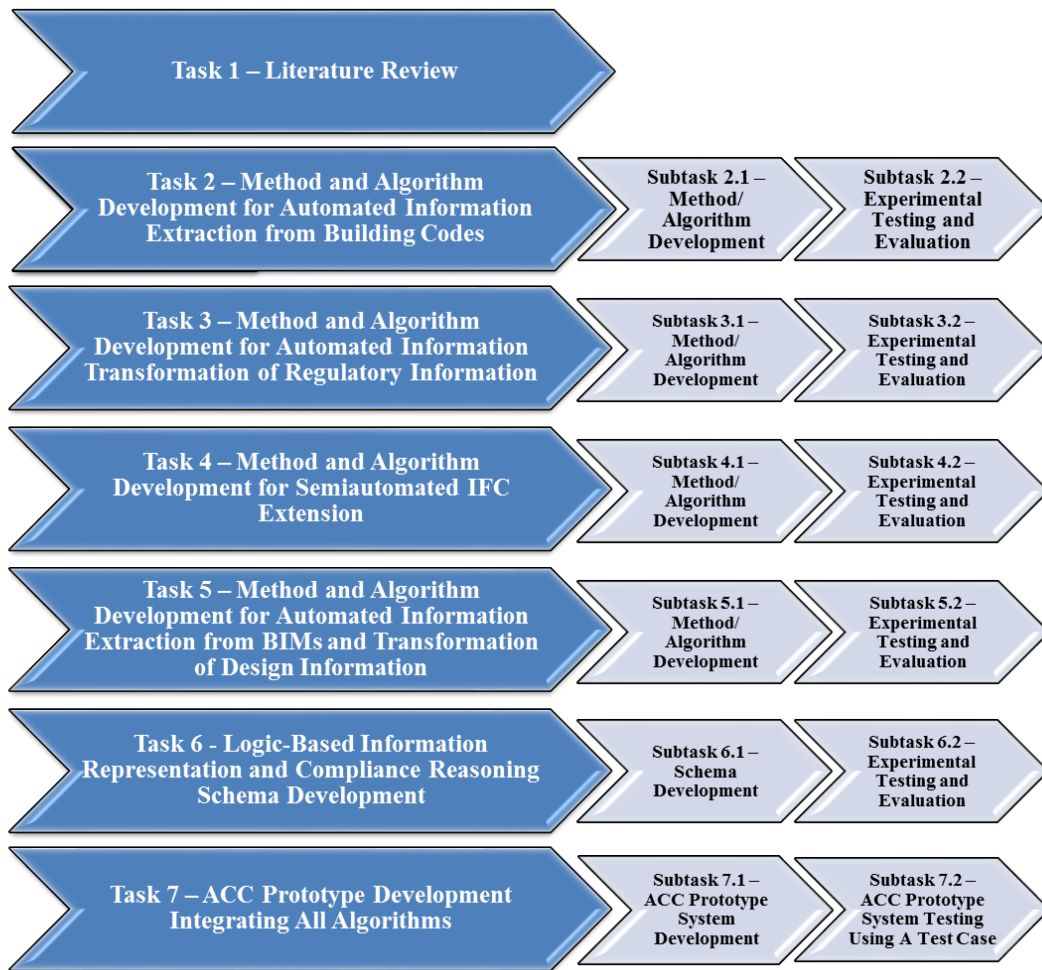


Figure 1.2 Research Methodology and Tasks

### 1.7.1 Task 1 – Literature Review

A literature review was conducted in six main research fields that are related to the scope of the dissertation: automated compliance checking in the construction domain, NLP, information extraction from BIMs, automated reasoning, semantic modeling and semantic-based NLP and reasoning, and machine learning algorithms. Concepts, methods/techniques, and tools/systems in these fields were analytically-reviewed, as follows:

- For automated compliance checking in the construction domain, the literature review

focused on reviewing existing efforts in terms of: (1) existing ACC efforts in academia and industry, (2) types of approaches utilized in the existing ACC efforts, (3) state of the art and gaps in existing ACC efforts; and (4) the need for automated rule extraction.

- For NLP, the literature review focused on: (1) basic concepts and techniques in NLP, (2) different types and levels of NLP, (3) grammars and theories that support NLP, (4) textual information extraction, (5) previous NLP work in the construction domain, and (6) evaluation methods for NLP tasks.
- For information extraction from BIMs, the literature review focused on: (1) data schema for BIMs, with a focus on Industry Foundation Classes (IFC), (2) different types of approaches for information extraction from BIMs, and (3) information requirements for information extraction from IFC-based BIMs.
- For automated reasoning, the literature review focused on: (1) different types of formally-defined logic and their advantages and limitations, (2) logic programming, especially Prolog, which is the most widely-used logic programming language, and (3) existing efforts in conducting automated reasoning in the construction domain.
- For semantic modeling and semantic-based NLP and reasoning, the literature review focused on: (1) the methods for ontology development, (2) previous ontological modeling efforts in the construction domain, (3) semantic information processing tasks in NLP, (4) the relation between semantic modeling and building information modeling, and (5) role

of semantic modeling in automated reasoning.

- For machine learning algorithms, the literature review focused on the main types of machine learning algorithms and their characteristics.

## **1.7.2 Task 2 – Method and Algorithm Development for Automated Information Extraction from Building Codes**

This task aimed at developing a semantic, rule-based NLP method and algorithm for automatically extracting regulatory information from building codes for supporting ACC in the construction domain.

### **1.7.2.1 Subtask 2.1 – Method/Algorithm Development**

This task focused on experimenting with different combinations of semantic NLP techniques to develop an information extraction method and algorithm that could achieve sufficient performance. The extracted information was represented in a domain-specific, computer-understandable format which is referred to as “semantic tuples” hereafter. There are two main types of approaches taken in NLP: a rule-based approach and a machine-learning (ML)-based approach. A rule-based approach utilizes human knowledge or heuristics in the development of the rules that are used for various language processing purposes. An ML-based approach applies machine learning algorithms (e.g., support vector machines, Naive Bayes, neural networks) on large volume of data for the training/development of NLP models (e.g., classifiers in the case of text classification) to

achieve the desired language processing objectives. A rule-based approach was taken in this dissertation, because of its expected higher performance in comparison to ML-based approaches when extracting information for a specific domain. Example NLP techniques used include morphological analysis, Part-of-speech (POS) tagging, pattern matching, etc. Two types of NLP methods were comparatively experimented with: syntactic NLP and semantic NLP. General architecture for text engineering (GATE) platform and tools (Cunningham et al. 2011), and Java programming language were utilized to implement various semantic NLP techniques.

#### 1.7.2.2 Subtask 2.2 – Experimental Testing and Evaluation

This task focused on testing the developed method and algorithm experimentally using well-established information extraction evaluation criteria – precision, recall, and F1-measure. Precision, here, is defined as the ratio of the number of correctly extracted information elements over the total number of information elements extracted. Recall, here, is defined as the ratio of the number of correctly extracted information elements over the total number of information elements that should be extracted. F1-measure is the harmonic mean of precision and recall. These measures were calculated based on a comparison of experimental results with a manually-developed gold standard, for information extracted from a randomly-selected chapter from an International Building Code.

### **1.7.3 Task 3 – Method and Algorithm Development for Automated Information Transformation of Regulatory Information**

This task aimed at developing a semantic, rule-based method and algorithm for automatically transforming the extracted regulatory information into logic rules for supporting ACC in the construction domain.

#### **1.7.3.1 Subtask 3.1 – Method/Algorithm Development**

This task focused on developing a rule-based information transformation method and algorithm. Two types of rules were used in information transformation: (1) semantic mapping rules, and (2) conflict resolutions rules. Semantic mapping rules define how to process the information instances according to their semantic meaning. The semantic meaning of each information instance is defined by the concept or relation it is associated with. Conflict resolution rules resolve conflicts in the transformation between different information elements. Python programming language was utilized to implement the information transformation method and algorithm.

#### **1.7.3.2 Subtask 3.2 – Experimental Testing and Evaluation**

This task focused on testing the developed method and algorithm experimentally using well-established criteria – precision, recall, and F1-measure. Precision, here, is defined as the ratio of the number of correctly generated logic clause elements over the total number of logic clause elements generated. Recall, here, is defined as the ratio of the number of

correctly generated logic clause elements over the total number of logic clause elements that should be generated. F1-measure is the harmonic mean of precision and recall. These measures were calculated based on a comparison of experimental results with a manually-developed gold standard, for information transformed from the extracted information (represented in semantic tuples) to the logic rules.

#### **1.7.4 Task 4 – Method and Algorithm Development for Semiautomated IFC Extension**

This task aimed at developing a semantic, rule-based and machine learning-based method and algorithm for semiautomatically extending the IFC schema with concepts in building codes for supporting ACC in the construction domain.

##### **1.7.4.1 Subtask 4.1 – Method/Algorithm Development**

This task focused on experimenting with NLP techniques, semantic similarity-based techniques, and machine learning techniques to develop an algorithm that semiautomatically extends the IFC schema with regulatory concepts in building codes. The algorithm should provide automation of the following processes in a sequential manner: (1) the extraction of regulatory concepts from building codes; (2) the matching of extracted regulatory concepts to their most related IFC concepts; (3) the classification of relationships between regulatory concepts and their most related IFC concepts; and (4) the integration of the regulatory concepts into the IFC schema. User actions should be allowed to fix the possible errors in the automated processes.

#### 1.7.4.2 Subtask 4.2 – Experimental Testing and Evaluation

This task focused on testing the developed method and algorithm by testing each of its processes. Depending on the nature of the process being tested, evaluation is conducted using well-established information processing evaluation criteria – precision, recall, and F1-measure, or simple evaluation criteria – adoption rate. Precision, here, is defined as the ratio of the number of correctly processed information elements over the total number of information elements processed. Recall, here, is defined as the ratio of the number of correctly processed information elements over the total number of information elements that should be processed. F1-measure is the harmonic mean of precision and recall. Adoption rate, here, is defined as the number of automatically selected IFC concepts (most relevant to the extracted regulatory concepts) that were adopted divided by the total number of automatically selected IFC concepts. These measures were calculated based on a comparison of experimental results with a manually-developed gold standard, for information processed from building codes.

#### **1.7.5 Task 5 – Method and Algorithm Development for Automated Information Extraction from BIMs and Transformation of Design Information**

This task aimed at developing a semantic, rule-based method and algorithm for automatically extracting design information from IFC-based building information models (BIMs) and automatically transforming the extracted design information into logic facts for supporting ACC in the construction domain.

#### 1.7.5.1 Subtask 5.1 – Method/Algorithm Development

This task focused on experimenting with different combinations of semantic rule-based techniques and BIMs information processing techniques to develop an algorithm that extracts instances of all ACC-related concepts and relations from BIMs and transforms the extracted instances into logic facts. This task included two main processes: (1) extraction of ACC-related concepts and relations from BIMs into a tuple format, for intermediate representation; and (2) transformation of the extracted BIM information (in tuple format) into logic facts that could be directly used for automated reasoning, along with regulatory rules. Python and Java programming languages were utilized to implement the various semantic rule-based techniques and BIMs information access techniques.

#### 1.7.5.2 Subtask 5.2 – Experimental Testing and Evaluation

This task focused on testing the developed method and algorithm using well-established information extraction evaluation criteria – precision, recall, and F1-measure. Precision, here, is defined as the ratio of the number of correctly extracted information elements over the total number of information elements extracted. Recall, here, is defined as the ratio of the number of correctly extracted information elements over the total number of information elements that should be extracted. F1-measure is the harmonic mean of precision and recall. These measures were calculated based on a comparison of experimental results with a manually-developed gold standard, for information extracted from IFC-based BIMs.



## **1.7.6 Task 6 – Logic-Based Information Representation and Compliance Reasoning**

### **Schema Development**

This task aimed at developing a logic-based information representation and compliance reasoning schema for representing, both, regulatory information and design information to prepare for utilizing the inference-making capabilities of logic reasoners for supporting ACC in the construction domain.

#### 1.7.6.1 Subtask 6.1 – Schema Development

This task focused on developing a logic-based information representation and compliance reasoning schema that could utilize regulatory and design information to conduct automated reasoning. The schema should enable the use of logic reasoners to support the automated reasoning process. The result of automated reasoning would be an assessment of whether the design complies with the code or not, with an analysis of the violated provision/rule. The different types of elements/components (e.g., facts, rules) in logic clauses were studied to determine the essence of the schema – what elements/components should be used to represent what information (e.g., regulatory or design), and how to represent them. Alternative schema designs were tested for comparison purposes. Prolog logic programming language was utilized to encode the representation schema.

#### 1.7.6.2 Subtask 6.2 – Experimental Testing and Evaluation

This task focused on testing the developed information representation and compliance

reasoning schemas in noncompliance detection using well established criteria – precision, recall, and F1-measure. Precision, here, is defined as the ratio of the number of correctly-detected noncompliance instances over the total number of noncompliance instances detected. Recall, here, is defined as the ratio of the number of correctly-detected noncompliance instances over the total number of noncompliance instances that should be detected. F1-measure is the harmonic mean of precision and recall. These measures were calculated based on a comparison of experimental results with a manually-developed gold standard, for information instances from a BIM test case that were designed to be checked for compliance with an International Building Code chapter.

### **1.7.7 Task 7 – ACC Prototype Development Integrating All Algorithms**

This task aimed at integrating and implementing all developed methods and algorithms in one platform: a proof-of-concept prototype system that conducts ACC based on building code text (.txt file) and a BIM design model (.ifc file).

#### **1.7.7.1 Subtask 7.1 – ACC Prototype System Development**

Guided by the logic-based information representation and compliance reasoning schema, which utilizes the inference-making capabilities of formally-defined logic-based reasoners for the automated reasoning process, the algorithms and semantic models for automated information extraction, automated information transformation, and automated compliance reasoning were integrated into a unified system using Java programming language. The

inputs to the integrated system are building code text (.txt file) and a BIM design model (.ifc file). The output of the integrated system is a compliance report.

#### 1.7.7.2 Subtask 7.2 – ACC Prototype System Testing Using a Test Case

The proof-of-concept prototype system was tested using a manually-developed test case (gold standard). The test case included: (1) a selected chapter from the International Building Code in text format (.txt format), (2) a building information model containing design information (LOD 400) to be checked for compliance with the regulatory provisions in the selected chapter (.ifc format), and (3) a compliance report gold standard based on a manual comparison of the design information with the regulatory provisions in the selected chapter. Testing included comparing the automatically-generated compliance report by the prototype system to the manually-generated compliance report. Evaluation was conducted based on measures of precision, recall, and F1-measure. Precision, here, is the number of correctly-detected noncompliance instances over the total number of detected noncompliance instances. Recall, here, is the number of correctly-detected noncompliance instances over the total number of noncompliance instances that should be detected. F1-measure is the harmonic mean of precision and recall.

## **1.8 Contribution and Significance**

### **1.8.1 Intellectual Merit**

This research explores a new approach to automated code compliance checking in the

construction domain. The contribution of the research lies in the following points.

- Domain-specific, semantic NLP methods for automatically extracting regulatory provisions from textual codes are offered that can: (1) help capture domain-specific meaning. Domain-specific semantics allow for analyzing complex sentences that would otherwise be too complex for automated information extraction and information transformation, recognizing domain-specific text meaning, and in turn improving performance of information extraction and information transformation. Supported by an ontology, NLP concepts and techniques (e.g., tokenization, sentence splitting, morphological analysis, part-of-speech tagging, phrase structure grammar) are used to facilitate textual document analysis and processing for extraction, transformation, and formalization of regulatory rules; and (2) achieve full sentence processing and information extraction (i.e., all terms of a sentence are processed), as opposed to partial sentence processing and information extraction (i.e., only specific terms/concepts are processed/extracted). Full sentence processing/understandability allows for a deeper level of NLP, namely natural language understanding. Deep NLP is achieved through a combination of domain knowledge (represented in the form of a domain ontology) and expert NLP knowledge (represented in the form of IE rules). This study is the first in the architectural, engineering, and construction (AEC) domain that addresses automated information processing (i.e., information extraction and information transformation) using

a semantically-deep NLP approach; and (3) achieve high performance in both information extraction and transformation, separately and in combination.

- Baseline semantic methods/algorithms for extracting and transforming information from textual building code documents were provided. Future research could use these methods/algorithms as a benchmark and build on this work by adapting the developed algorithms to extract and transform information from other types of construction documents (e.g., contract documents) or for different purposes (e.g., contract analysis). Compared with the author's initial efforts, future efforts in adapting the rules and/or algorithm should be significantly lower.
- A new IFC extension method is provided which: (1) objectively and semiautomatically extends the IFC schema with domain-specific concepts that are extracted from natural language documents; and (2) follows the representation convention of existing .ifc files, which enables compatibility between newly incorporated information and existing IFC-based BIM information.
- A new BIM information extraction and information transformation method is provided that enables direct flow of design information from .ifc files to logic representations. As a result, this method allows for direct extraction of IFC-represented data into logic facts. This enables information transfer between BIMs and logic programs. In addition to supporting ACC, the combined capabilities of building information modeling and logic

programming could allow for the use of BIM information in an intelligent way and could open the door to more utilization of building information modeling in various automated applications in the construction domain such as automated cost analysis, schedule analysis, and facility maintenance decision analysis.

- A new logic-based, semantic schema is provided for representing building code provisions and design information in a way that is generalized and flexible. The logic-based representation allows for using the powerful reasoning capabilities of automated logic reasoners. The semantic representation enables deep reasoning and facilitates human understandability and interpretability of the formal representation. The proposed information representation and compliance reasoning schema could be benchmarked to support other automated applications in the construction domain that would benefit from such representation such as automated contract analysis.
- A novel proof-of-concept prototype system for ACC of building design with building codes. Compared to the state-of-art ACC systems, this prototype system has the following advantages: (1) compared to the state-of-the-art ACC systems that require manual encoding of regulatory information, this prototype system automates the extraction and transformation of both regulatory information and design information; (2) the automation offered by this prototype system could improve the consistency of the code analysis (compared to manual reading and interpretation) and in turn could improve the consistency of the compliance checking process; and (3) this prototype system provides

logic-represented regulatory information and design information which could be leveraged in other types of reasoning systems to conduct more analyses of building designs and/or building codes in an automated way.

### **1.8.2 Broader Impact**

ACC could have significant benefits to the society by:

- Reducing the time and cost of compliance checking in the construction domain: Checking compliance with applicable laws and regulations has been costly and time-consuming to all relevant stakeholders. ACC is expected to enhance the efficiency of the process, and consequently reduce the associated time and cost.
- Improving the accuracy of compliance checking: Automating the compliance checking process is expected to reduce the errors of compliance checking by eliminating human-caused errors that may occur during manual checking.
- Supporting other applications of automated information processing in the construction domain: The application of this study could be extended to support automated information processing and analysis for many other applications and purposes, such as analysis of contract documents for the detection of inconsistencies, analysis of project documents and records for supporting claim analysis, analysis of daily site reports for supporting progress monitoring and project control, etc.

- Providing insights into a potentially formal representation of building regulations which could better support automated compliance checking than the status quo. This formal representation could provide guidance for writing future building codes in a way that facilitates automated compliance checking and could open a new direction of research in utilizing logic-based reasoning in the construction domain.



## **2 CHAPTER 2 – LITERATURE REVIEW**

This chapter describes the reviewed literature in the following research fields which are related to the dissertation scope of work: automated compliance checking (ACC) in the construction domain, Natural Language Processing (NLP), information extraction (IE) from building information models, automated reasoning, semantic modeling and information processing, and machine learning algorithms.

### **2.1 Automated Compliance Checking (ACC) in the Construction Domain**

#### **2.1.1 Previous ACC Efforts in the Construction Domain**

There have been significant research efforts to automate the compliance checking process, such as the checking of building envelope performance (Tan et al. 2007; Tan et al. 2010), fire code compliance (Delis and Delis 1995), facility accessibility code compliance (Han et al. 1997), requirements of safety and reliability of structures (Garrett and Fenves 1987), accessibility (Lau and Law 2004), egress, environmental protection, and energy conservation (FIC 2007), building design (Eastman et al. 2009), and construction inspection and quality control (Boukamp and Akinici 2007). Larger research and software development efforts for automated building code checking led by industry bodies/associations, software companies, and/or government organizations include Solibri Model Checker (Corke 2013), EPLAN/BIM and AutoCodes led by Fiatech (Fiatech 2011; 2015), CORENET led by the Singapore Ministry of National Development (Singapore Building and Construction Authority 2006),

REScheck and COMcheck led by the U.S. Department of Energy (US DOE 2011), SMARTcodes led by the International Code Council (ICC 2011), and Avolve Software (Avolve Software Corporation 2011).

Existing ACC efforts took various approaches. For example, Fenves et al. (1969) formalized the American Institute of Steel Construction (AISC) specifications into decision tables; Garrett and Fenves (1987) proposed a strategy to represent design standards using information networks and represent design component properties using data items for ACC of structural designs; Ding et al. (2006) proposed an approach to represent building codes using object-based rules and represent designs using an Industry Foundation Classes (IFC)-based internal model for ACC of accessibility regulations; Tan et al. (2010) proposed an approach to represent building codes and design regulations using decision tables and incorporate simulation results in building information models for ACC of building envelope design; the CORENET project of Singapore (Khemlani 2005) used an approach to represent design information using semantic objects in the FORNAX library (i.e., a C++ library) and represent regulatory rules using properties and functions in FORNAX objects for ACC of building control regulations, barrier free access, and fire code, etc.; and the SMARTcodes project (ICC 2011) of the International Code Council (ICC) used an approach to represent ICC codes in computer-processable tuple format and represent designs using an IFC-based model for ACC of designs with ICC codes, etc.

The most recent ACC project in the U.S. is the AutoCodes project by Fiatch. The

AutoCodes Project by Fiotech aims to develop an open-source and nonproprietary ruleset library for checking code compliance on building information models (Fiotech 2011). The AutoCodes project includes three phases with deliverables: (1) The deliverables for Phase I (March 2011 to January 2012) are open source and nonproprietary rulesets for accessibility and egress compliance checking (this scope overlaps with ICC's SMARTcodes project plan declared in October 2011); (2) The deliverables for Phase II (March 2012 to October 2014) are open source and nonproprietary rulesets for fire and life safety and/or mechanical and engineering model-based building codes; and (3) The deliverable for Phase III (called future phases in Fiotech's plan) are open source and nonproprietary rulesets for all rules needed in the current compliance checking process (Fiotech 2011). The concluding report of Phase I (Fiotech 2012) reported five main findings on building information model authoring, jurisdiction reorganization and process transformation, jurisdiction reporting need, education of jurisdiction officials, and involvement improvement of jurisdiction officials, respectively. However, progress on the open-source and nonproprietary rulesets for accessibility and egress was not reported. Phase II of AutoCodes project was said to be making "considerable progress by applying innovative technology to enable a digital review process, including automated code checking of building information models (BIMs)." (Fiotech 2014). Still, the progress on rulesets development has not been explicitly reported. A recent interview with AutoCodes project officials in April 2015 indicates that the AutoCodes project now shifted focus to defining a modeling matrix (i.e., an information template) for guiding design firms to

design models that can be automatically checked with building codes, away from the earlier goal of creating rulesets. Instead, another Fiotech project called “U.S. Local Codes in the Cloud” project was launched, which aimed to hard-code all local building codes and amendments into a database in the cloud and provide annual fee-based subscription to users. The “U.S. Local Codes in the Cloud” project is now planned to provide rulesets database to AutoCodes project (Fiotech 2015). However, if the subscription is provided for an annual fee, then the rulesets would not be entirely open-source and nonproprietary as originally intended. The use of a fee, in the author’s view, is related to the big cost needed for manual rule encoding. As such, similar to other ACC efforts (e.g., the SMARTcodes project), the main gap associated with the AutoCodes project is the need for manual rule encoding which is time-consuming, costly, and error-prone.

### **2.1.2 Limitations of Previous ACC Efforts in the Construction Domain**

Previous research efforts have undoubtedly paved the way for ACC in the AEC industry. However, these efforts are limited in their automation and reasoning capabilities (Zhong et al. 2012).

Existing ACC systems require manual effort for extracting requirements from textual regulatory documents (e.g., codes) and encoding these requirements in a computer-processable rule format. Rules are either hard-coded into the developed systems or hand-coded as a rule database or set of files. Accordingly, all existing representations of building regulations need to be manually-updated to reflect the current status of the source

documents that are subject to constant changes (Dimyadi and Amor 2013). This makes the automation of compliance checking only semiautomated. For example, the software tools developed by OptaSoft for ACC with ICC codes need major manual data entry and navigation (OptaSoft 2014). Similarly, Solibri Model Checker implements a ruleset manager to manage a set of regulatory rules that are built-in (i.e., currently these are rules for checking accessibility based on the International Organization for Standardization (ISO) accessibility code and exit path distance based on fire code). Rules could be adjusted by tuning preset parameters. But the addition of new rules have to be conducted by Solibri experts (Eastman et al. 2009; Corke 2013).

In terms of reasoning, most of the existing ACC efforts utilized proprietary (thus hidden) information representation and reasoning mechanisms. For example, both OptaSoft and Solibri use proprietary information representation and reasoning mechanism, which are not easily understandable and testable by users. In addition, the use of various information representation methods leads to difficulty in interoperability. According to a conservative estimate by the national institute of standards and technology (NIST) (Gallaher et al. 2004), the lack of interoperability in the U.S. capital facilities industry costs \$15.8 billion per year. Although few proposals have been made such as the requirement, applies, select, and exception (RASE) representation (Hjelseth and Nisbet 2011) and the semantic resource description framework (RDF) annotations (Yurchyshyna and Zarli 2009), there is a lack of a generalized and flexible schema to allow for deep representation of all information (i.e., all

concepts and relations) in a regulatory provision regardless of the type, length, and complexity of the provision (sentence). The representation needs to be non-proprietary, non-hidden, and user-understandable and testable (Garrett and Palmer 2014).

### **2.1.3 Need for Automated Rule Extraction**

The process of manual rule extraction and encoding (referred to as manual rule encoding hereafter) is time-consuming, costly, and error-prone (Selvi et al. 2015; Marcinczuk and Ptak 2012). First, manual rule encoding is time-consuming. For example, the SMARTcodes project by ICC started encoding the International Energy Conservation Code (IECC) 2006 in a “smart” format in 2006 and only finished the encoding of envelope and lighting provisions of the IECC 2006 (less than 32% of the 264 provisions in the IECC 2006) by October 2007 (ICC 2007). The remaining provisions of the IECC 2006 were still not completed by January 2011 (ICC 2011). Finally, ICC ended up joining efforts with Fiatch, as part of the AutoCodes Project, due to the difficulty and time-consuming nature of manual rule encoding. Furthermore, new editions of the ICC codes are published every three years. In the middle of this three-year cycle, a supplement to the then current edition of the ICC Codes that contains all the approved changes to the code during the first 18-month code change cycle is published (ICC 2012). Accumulating changes in a 36-month cycle could lead to a large number of changes in provisions in each new edition. For example, from IBC 2003 to IBC 2006, more than 450 provisions were changed (ICC 2012). Thus, it is difficult for manual rule encoding to catch up with the rate of code updates. This is compounded by two more facts: (a) IECC is

only one code out of 13 in the ICC codes family. New versions for all 13 codes are published at the same rate. Therefore, all 13 codes will need manual encoding effort to a similar extent (effort may vary depending on the size and complexity of each code); and (b) there are many other referenced standards in each code, which will also need manual encoding efforts upon the publishing of their new versions. For example, IECC 2006 referenced 50 other standards.

Second, manual rule encoding is costly, because manual rule encoding requires expertise in both the interpretation of rules and the writing of rules in the specific formats/languages used for encoding. Getting knowledge from an expert into a computable rule is currently a long, labor intensive, complex, and costly process (Bell et al. 2009). For example, in Solibri Model Checker, rules that were not in the built-in rule configuration tool need to be encoded by experts from Solibri, as well as domain experts with knowledge about the domain of the rules (Bell et al. 2009). Because manual rule encoding is costly, Solibri provides customized rule encoding as a charged service (Corke 2013).

Third, manual rule encoding is error prone. Construction projects must comply with a multitude of regulations, which increases the complexity of manual rule encoding. In one dimension, these regulations come from different jurisdictional levels (i.e., federal level, state level, and local level). In another dimension, these regulations come from different domains (e.g., building code requirements, electrical requirements, and fire protection requirements). The encoding of each regulation needs the interpretation of rules by experts with knowledge at that specific level and in that specific domain. For example, as shown in Table 2.1, a

construction project in the city of Champaign, state of Illinois must comply with more than twenty regulations (City of Champaign 2015). Each regulation could contain hundreds to thousands of provisions. This level of complexity makes any manual effort for processing the provisional information in the regulations quite prone to errors, including rule interpretation and encoding.

To better support ACC, an automated rule extraction and encoding method is needed to reduce the time, cost, and errors in the rule extraction and encoding tasks.



Table 2.1 Regulations for a Construction Project in Champaign, Illinois

	Federal level	State level	Local level
Regulations to comply to for a construction project in Champaign, Illinois	2009 International Building Code (IBC) 2009 IBC Amendments	2004 Illinois Plumbing Code (ILPC) 2004 ILPC Amendments	Champaign-Urbana Public Health District Regulations
	2009 International Residential Code (IRC) 2009 IRC Amendments	1997 Illinois Accessibility Code	Urbana-Champaign Sanitary District Regulations
	2009 International Fire Code (IFC) 2009 IFC Amendments	2012 Illinois Energy Conservation Code (IECC) 2012 IECC Amendments	
	2009 International Mechanical Code (IMC) 2009 IMC Amendments	Illinois American Water Regulations	
	2009 International Fuel Gas Code (IFGC) 2009 IFGC Amendments	Illinois Dept. of Transportation Regulations	
	2008 National Electrical Code (NEC) 2008 NEC Amendments	Illinois Emergency Management Agency Radon Program Regulations	
	US Environmental Protection Agency (EPA) Regulations	Illinois EPA Asbestos Program Regulations	
	US EPA Lead-Based Paint Regulations	Illinois State Fire Marshal Regulations	
	EPA Contractor Lead Safety Brochure		
	2010 Americans with Disabilities Act (ADA)		
	1988 Federal Fair Housing Act (FHA)		
	US EPA Asbestos Regulations		
	Occupational Health and Safety Regulations (OHSA)		

## **2.2 Natural Language Processing (NLP)**

### **2.2.1 Overview of Natural Language Processing**

Natural language processing (NLP) is a field utilizing artificial intelligence to enable computers to understand and process natural language text (and speech) in a human-like manner (Cherpas 1992; Marquez 2000). Example NLP tasks include text summarization, machine translation, handwriting recognition, speech recognition, semantic role labeling, and information extraction (IE) (Marquez 2000; McCallum et al. 2005). NLP, typically, is composed of a set of well-defined tasks. Examples are tokenization, part-of-speech (POS) tagging, morphological analysis, named entity recognition, and co-reference resolution. (Patwardhan 2010).

NLP techniques are commonly classified into machine learning-based approaches and rule-based approaches (using human-developed rules). A machine learning-based approach applies machine learning algorithms (e.g., support vector machines, Naive Bayes, neural networks) on large volume of data for training/development of NLP models (e.g., classifiers in the case of text classification) to achieve the desired language processing objectives. It relies heavily on the training/development data and the training rationale is usually difficult to be understood/interpreted by human intuitively (Deokar and Sen 2010; Pradhan et al. 2004). The main essence of a rule-based approach is the utilization of human knowledge or heuristics in the development of the rules used for various language processing purposes. A

rule-based method is, therefore, less-dependent on a development/training data set which allows the method to better generalize to other corpora (Saric et al. 2005; Manning et al. 2009; Kim et al. 2010). Rule-based language processing tends to show equal or better performance (in terms of precision and recall) than machine learning-based processing (Abney 1997; Pilon et al. 2014). However, the initial manual effort that is needed for the development of the rules might pose concerns on development efficiency. But the developed rules could be generalized and could be highly reusable (Abney 1997; Califf and Mooney 2003), which would require much less manual effort in adapting the developed rules for use in processing other text in similar domains. So, if the application domain is well-defined and relatively small (or at least finite), rule-based language processing is, usually, a more suitable approach than machine learning-based processing.

NLP may be classified into two main levels: shallow NLP and deep NLP. Shallow NLP focuses on partial analysis of sentences. Example tasks in shallow NLP are part-of-speech (POS) tagging and text chunking, which aim at assigning part-of-speech labels (e.g., noun, verb, adjective) to each word of a sentence and dividing sentences into meaningful chunks, respectively. Deep NLP aims at achieving complete analysis and processing of sentences. It, thus, involves handling of very fine-grained aspects of languages such as anaphora resolution (i.e., finding what “she,” “he,” “they,” and “it” are referring to in a sentence) and quantifier scope resolution (i.e., resolving scope ambiguities caused by terms like “every,” “some,” “a,” “many,” or “a few” in a sentence, such as in the sentence “Every monkey climbed a tree”).

An example task of deep NLP is complete parsing of sentences, which requires complete understanding of the entire meaning carried by the sentences (Zouaq 2011). With decades of development, the general picture in NLP is that it is possible to do shallow NLP at large-scale and with reasonable accuracy, but deeper NLP is generally hard, because of the need for knowledge representation and inference-making to accurately understand the meaning of a natural language text (Wen et al. 2012; Jones 1997).

### **2.2.2 Rule-Based NLP Using Pattern-Matching-Based Rules**

Rule-based NLP uses manually coded rules for text processing. These rules are iteratively constructed and refined to improve the accuracy of text processing. ML-based NLP uses ML algorithms to train text processing models based on the text features of a given training text (Tierney 2012). Rule-based NLP tends to show better text processing performance (in terms of precision and recall) but requires greater human effort.

Pattern-matching-based rules are widely-used in NLP tasks such as POS tagging (Abney 1997; Yin and Fan 2013), information extraction (Califf and Mooney 2003), and text understanding (Goh et al. 2006). The idea of pattern-matching-based rules is to define a set of results when the matching of a pattern of a specific sequence (or structure like a tree) of elements (e.g., characters, tokens, symbols, terms, concepts) occurs. Pattern-matching-based rules have a variety of implementations tailored to different purposes and domains. But, they all share the same rule schema of “if *pattern* then *result*” or the mapping of “from *pattern* to

*result.*” For example, in the rules for information extraction from textual documents, the result is the recognition and extraction of information element instances. In the rules for information transformation from extracted information element instances, the result is the transformation of information instances into new representations.

### **2.2.3 Phrase Structure Grammar**

When processing natural languages, the expert’s (researcher or system developer) understanding of the language structure is essential. Chomsky’s transformational grammar assists in such understanding; it dominated linguistic studies in the second half of the 20<sup>th</sup> century. His transformational grammar stimulated a significant amount of theoretical and computational research studies in language processing. Phrase structure grammar (PSG) is such a transformational grammar; it is defined by “a finite vocabulary (alphabet)  $V_p$ , a finite set  $\Sigma$  of initial strings in  $V_p$ , and a finite set  $F$  of rules of the form:  $X \rightarrow Y$ , where  $X$  and  $Y$  are strings in  $V_p$ ” (Chomsky 1956). An important characteristic of a PSG is that it singles out and encodes the most important recursive structure and syntactic constituency of a sentence (Levine and Meurers 2006). Using PSG, complex syntactic features of a language could be represented by a few or even just one simple symbol. This advantage makes PSG a potentially powerful technique for encoding complex language structures. Context free grammar (CFG) is a more restricted form of PSG. The restriction of CFG beyond general PSG is that the left-hand side of a generative rule ( $L$  in  $L \rightarrow R$ ) has to be a single non-terminal (i.e., a symbol that could be further broken down) (Joshi 1991). This restriction further

simplifies the representation of complex language structures. However, when relations of words that are far from each other become important in language analysis, the representation using CFG becomes inefficient due to complexity in such use. Dependency grammar (DG), in this case, is more suitable to use. The basic construct for DG is the pairwise relation between two words. In each relation, one of the words is called “head” and the other is called “dependent.” Parsing with DG is more straightforward than parsing with PSG, because it only needs to connect existing nodes (i.e., the words in a sentence), while more intermediate-level nodes need to be created as in parsing with PSG. Thus, parsing with DG is simpler, but less expressive (Covington 2001).

#### **2.2.4 Information Extraction**

Information extraction is a subfield/subdiscipline of NLP. Information extraction aims at extracting facts and structured information from unstructured natural texts, and filling them in pre-defined information templates (Riloff and Lorenzen 1999). It is the “identification, and consequent or concurrent classification and structuring into semantic classes, of specific information found in unstructured data sources, such as natural language text, making the information more suitable for information processing tasks” (Moens 2006).

Nowadays, as a result of the fast-growing amount of data/information, information extraction is in great need. It is especially required where manual information processing would be too time-demanding and/or too complex (e.g., extracting information – such as product features

and consumer opinion – from the enormous amount of online review data) (Popescu 2007). Information extraction applications have been widely used in many domains (Liddy 2003). Typical applications of information extraction include text mining (Humphreys et al. 2000; Müller et al. 2004), semantic annotation (Liu and Singh 2003), question answering (Banko et al. 2002; Magnini et al. 2002), review and opinion mining (Turney 2002; Turney and Littman 2003; Popescu 2007; Kim and Hovy 2004), decision support (Gupta and Kochenderfer 2004), rich information retrieval and exploration (Hauptmann et al. 2003; Wagner et al. 2006), etc. Many information extraction systems or approaches have been developed. For example, Plake et al. (2005) described a general approach to the task of information extraction from free text and proposed methods for learning syntax patterns automatically from annotated corpora; Buitelaar et al. (2008) presented the design, implementation, and evaluation of the SmartWeb ontology-based annotation (SOBA), a system for ontology-based information extraction from heterogeneous data resources, including plain text, tables and image captions; Roth-Berghofer et al. (2010) showed how to use SCOOBIE (i.e., an ontology-based information extraction system) for generating cases from texts; and Wimalasuriya and Dou (2010) developed a comprehensive component-based approach for information extraction that promotes reuse.

Information extraction approaches could be generally categorized into two types: syntactic-based and semantic-based. The Little Oxford dictionary (1986) defines syntactic knowledge as “the grammatical arrangement of words/rules or analysis of it” and semantic

knowledge as “the meaning in language.” Semantic information extraction can be achieved through the use of ontologies, which are utilized to represent domain knowledge. Ontology-based information extraction and several example systems were described in (Wimalasuriya and Dou 2010). Ontology-based information extraction is expected to have better performance in comparison to syntactic-based information extraction, because domain knowledge represented in an ontology could help to distinguish and disambiguate domain-specific terms and meanings (Saggion et al. 2007; Soysal et al. 2010).

The state-of-the-art semantic information extraction studies have four major focuses: named entity extraction, attribute extraction, relation extraction, and event extraction. Named entity extraction, attribute extraction, and relation extraction aim at extracting instances of a single concept (e.g., named entity) or of two related concepts (Ling and Weld 2012; Pasca 2011; Wang et al. 2010). Event extraction aims at extracting instances of multiple concepts (Patwardhan 2010). From this perspective, a great need exists in researching complex information extraction from text, which should go beyond extraction of information elements with pre-defined, fixed number of concepts/relations (e.g., in a terrorist event case, it is pre-defined that “victim” is associated with only one concept) to achieve an interpretation level information extraction (e.g., information element with a varying number of multiple concepts/relations depending on the sentence being extracted from).

An essential technique for information extraction is pattern matching, which defines the actions to take when a specific matching pattern is met in the text. Matching patterns are the



patterns expressed by sequences/structures of features. A variety of features could be utilized in matching patterns such as tokens, POS tags, text structural information, and semantic information. Typically text features are classified into two types: syntactic features and semantic features. Syntactic features, such as POS tags, are widely-used for information extraction, such as in Afrin (2001). Semantic features benefit information extraction tasks beyond solely using syntactic features because they express domain-specific meaning/knowledge, such as in Soysal et al. (2010).

### **2.2.5 Previous NLP Efforts in the Construction Domain**

In the construction domain, a number of important research efforts utilized NLP techniques. For example, Caldas and Soibelman (2003) conducted ML-based text classification of construction documents. However, only a few of these efforts conducted some type/level of information extraction, such as Abuzir and Abuzir (2002) and Al Qady and Kandil (2010). Al Qady and Kandil (2010) used shallow parsers to extract concepts and relations from construction contracts. In Al Qady and Kandil (2010), (1) the extraction is only based on syntactic features produced by shallow parsing; and (2) information recognition is based on specific types of phrases and their roles (produced by shallow parsing) [e.g., noun phrase (NP) segment and its role SUBJ (i.e., subject)], which allows for extracting relations between concepts. Abuzir and Abuzir (2002) used information extraction techniques to extract terms and relations from HyperText Markup Language (HTML) documents for constructing a civil engineering thesaurus. In Abuzir and Abuzir (2002), (1) the extraction uses HTML-based

document structure features [including title tags, heading tags, and uniform resource identifiers (URLs)] and simple lexical syntactic features; and (2) because the main purpose of the extraction is thesaurus construction, their information extraction focuses on extracting terms.

### **2.2.6 Evaluation of NLP Methods**

Precision, recall, and F1-measure are the most widely used evaluation metrics in many NLP tasks such as information retrieval (Huang et al. 2005; Manicassamy et al. 2012), information extraction (Guo et al. 2012), text classification (Bui and Zeng-Treitler 2014), and machine translation (Rathod 2014). Precision is defined as the number of correctly processed information elements divided by the total number of information elements processed. Recall is defined as the number of correctly processed information elements divided by the total number of information elements that should have been processed. A trade-off exists between precision and recall, using either indicator alone is not sufficient. F1-measure is defined as a weighted combination (harmonic mean) of precision and recall (Makhoul et al. 1999).

The above definition of these three metrics could be used to evaluate many NLP tasks by replacing the meaning of “processed” with the specific NLP task evaluated. For example, in the task of information retrieval, precision is defined as the number of correctly retrieved documents (i.e., relevant documents) divided by the total number of retrieved documents. Recall is defined as the number of correctly retrieved documents divided by the total number of documents that should have been retrieved (Van Rijsbergen 1979).

The definitions of precision, recall, and F1-measure are shown in Equations (2.1) to (2.3) (Olson and Delen 2008). A true positive (TP) is a correctly processed information element. A false positive (FP) is an incorrectly processed information element. A false negative (FN) is an information element that should have been processed but was not.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.2)$$

$$\text{F1-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.3)$$

When evaluating an NLP task using precision, recall, and F1-measure, a gold standard is typically used for comparison with the output results of the NLP task (Poibeau and Messiant 2008). A gold standard is a set of manually-developed standard outputs of an NLP task on a specific set of testing data (i.e., a corpus) (Wissler et al. 2014; Deleger et al. 2014). A gold standard is typically developed by manual effort (Deleger et al. 2014; Zhai et al. 2013). Depending on the intended use of a gold standard, the gold standard development could be conducted in various ways, and the researchers typically describe the steps taken in their gold standard development (e.g., Bernier-Colborne 2012; Al Qady and Kandil 2010). Because the development of a gold standard requires manual interpretation and/or understanding, to ensure the correctness and objectivity of the developed gold standard, typically, efforts from more than one person are needed so that their interpretation and/or understanding could be checked with each other (Wiebe et al. 2005). The agreement on the interpretation and/or

understanding among a group of individuals is measured by inter-annotator agreement. Different measures could be used to report inter-annotator agreement for different types of NLP tasks. For example, Kappa is the best metric for reporting inter-annotator agreement of text classification tasks (Carletta 1996; Cunningham et al. 2011). Precision, recall, and F1-measure, on the other hand, are typically used for reporting inter-annotator agreement of information extraction tasks by treating one of the annotations as a gold standard (Cunningham et al. 2011).

To test the statistical significance of the measures of precision, recall, and F1-measure, standard statistical testing techniques could be used to evaluate the confidence intervals of the measures (Meystre and Haug 2005; Goutte and Gaussier 2005). Because developing a gold standard for an NLP task could be very costly and time-consuming, it is common in NLP evaluation that only one testing set is used. In this case, the confidence intervals for the precision, recall, and F1-measure could be calculated using the confidence interval calculation method for a single proportion (Tetreault and Chodorow 2008). Example confidence interval calculation methods for a single proportion include simple asymptotic method without continuity correction, asymptotic method with continuity correction, and Wilson score method without continuity correction (Newcombe 1998). Wilson score method without continuity correction is simpler and more plausible comparing to other methods (Newcombe 1998). In the Wilson score method without continuity correction (Wilson 1927), the equation for calculating the confidence interval  $p$  for a single proportion  $p_0$  in a

population of size  $n$  is:

$$p = \frac{p_0 + t/2}{1+t} \pm \frac{\sqrt{p_0 q_0 t + t^2/4}}{1+t} \quad (2.4)$$

In Equation (2.4),  $p_0$  is the proportion of instances with a certain phenomenon from a population of size  $n$ ,  $q_0$  is  $1 - p_0$ ,  $t$  is  $\lambda^2/n$ , and  $\lambda$  is the critical value for the corresponding confidence level. In estimating the confidence interval for precision in information extraction,  $n$  is the number of extracted information instances and  $p_0$  is the ratio of the number of correctly extracted information instances over the total number of information instances extracted. For example, if 509 information instances are extracted and 493 information instances are correctly extracted, then  $n$  is 509,  $p_0$  is 96.9%, and  $\lambda$  is 1.96 at 95% confidence level. Thus, the confidence interval at the confidence level of 95% is calculated to be [95.0%, 98.1%] and reported together with the precision as 96.9% (95% confidence interval [95.0%, 98.1%]). In estimating the confidence interval for recall in information extraction,  $n$  is the number of information instances that should be extracted and  $p_0$  is the ratio of the number of correctly extracted information instances over the total number of information instances that should be extracted. For example, if 522 information instances should be extracted and 493 information instances are correctly extracted, then  $n$  is 522,  $p_0$  is 94.4%, and  $\lambda$  is 1.96 at 95% confidence level. Thus the confidence interval at the confidence level of 95% is calculated to be [92.1%, 96.1%] and reported together with the recall as 94.4% (95% confidence interval [92.1%, 96.1%]). The lower bound (or upper bound) of F1-measure are calculated using the lower bounds (or upper bounds) of precision and recall, using the same equation for

calculating F1-measure.

## **2.3 Information Extraction from Building Information Models**

### **2.3.1 Building Information Modeling and Industry Foundation Classes (IFC)**

According to the National Building Information Model Standard Project Committee (National Institute of Building Sciences 2014), a building information model (BIM) is “a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition.” BIM is believed to improve interoperability through structured information and coordinated information flow during a building life cycle and between different disciplines (Hamil 2012). However, although BIM is intended to be fully interoperable, in reality different BIM softwares and platforms are not yet realizing full compatibility and seamless information exchange hitherto, which prevents BIM from realizing its full potential (Young et al. 2009).

Standardization is a primary way to improve interoperability. The current main standardization efforts in BIM include Industry Foundation Classes (IFC) and the CIMSteel Integration Standards (CIS/2) (Isikdag et al. 2007). The IFC represents the main data model to describe building and construction industry data. The IFC schema specifications are written using the EXPRESS data definition language (ISO 10303-11 by the ISO TC 184/SC4 committee) (BuildingSmart 2014). The IFC schema is the data exchange standard to facilitate

interoperability in the construction industry (BuildingSmart 2014). The CIS/2 is a product model and data exchange file format for structural steel project information (AISC 2014). Both IFC and CIS/2 models are defined using Standard for Exchange of Product model data (STEP) description methods, which is the official “Standard for Exchange of Product model data” – ISO 10303. In contrast to CIS/2 which is focusing on modeling information of structural steel framework, IFC schema is designed to cover all subdomains and phases of the building and construction industry. Thus, IFC attracted more attention in BIM research. IFC schema is neutral and platform independent. It is defined using the STEP description method, which is the official data description standard ISO 10303. The IFC schema is registered as ISO/PAS 16739 and is registered as an official international standard ISO 16739:2013.

In the IFC schema, concepts are represented by entities. Non-hierarchical relations between concepts are represented through attributes of entities. There are three types of attributes: explicit attribute, derived attribute, and inversed attribute. Explicit attribute is an attribute whose value is directly visible in a STEP file. Derived attribute is an attribute whose value can be computed from an expression, which may refer to other attributes and use functions. Inversed attribute is just a name representing the inversed direction of a relationship represented by an explicit attribute. In the IFC schema, an attribute adds a property to an entity or relates an entity to another entity. For example, the entity “IfcOrganization” in the IFC schema has an explicit attribute “addresses,” which relates an organization to one or more “addresses.” The use of derived attribute in an IFC schema is mainly in defining

attributes of geometric representations and operations. For example, “IfcDimensionCount” is a derived attribute of “IfcCurveBoundedPlane,” which is derived from the dimension of the basis surface. In the IFC schema, the entity “IfcRelationship” is a direct subtype of “IfcRoot.” Subtypes of “IfcRelationship,” which follow the format of “IfcRel...” are designed to be designated to represent relationships. For example, “IfcRelCoversSpaces” represents the relationship between one or more coverings and a space that those coverings cover. It has two attributes: the “RelatedSpace” and the “RelatedCoverings.” The value of the “RelatedSpace” attribute represents an “IfcSpace,” and the value of the “RelatedCoverings” attribute represents a set of “IfcCoverings” that cover the “IfcSpace” (BuildingSmart 2014). The reader should note that any entity in the IFC schema could actually be used like an “IfcRel...” entity, and the creation of “IfcRel...” entities grouped under “IfcRelationship” is mainly for organizational purpose.

The specifications of the IFC schema are written using the EXPRESS data definition language (BuildingSmart 2014). EXPRESS is an ISO standard product data modeling language (ISO 2004). The EXPRESS data definition language has the following five main data types: simple data types, aggregation data types, named data types, constructed data types, and generalized data types. The simple data types include seven data types: number, real, integer, string, Boolean, logic, and binary. The aggregation data types include four data types: array, list, bag, and set. The named data types include the entity data type and the defined data type. The constructed data types include the enumeration data type and the select



data type (ISO 2004). The generalized data types are not used in the IFC schema and are not of interest here. Among these data types, entity is the most important data type used in the IFC schema. The entity data type in EXPRESS follows an object-oriented data structure to represent a concept. Its use in existing IFC schemas (e.g., IFC\_2X3\_TC1) is extended to represent a relation as well (as mentioned earlier). The enumeration data type enumerates predefined values for a concept category using simple strings. For example, “swinging,” “double acting,” “sliding,” “folding,” “revolving,” and “rollingup” enumerate the predefined values for the concept category “door panel operation.” The defined data type defines customized data types by adding constraints to an existing data type. For example, “positive integer” is a defined data type by adding the “greater than zero” constraint to the “integer” data type. The select data type defines a selection among different data types or entity types. For example, a select data type “shell” may define a selection among two entity types “closed shell” or “open shell.” The aggregation data type defines an ordered or unordered set of any data type using list, set, bag, or array. For example, an aggregation data type “name” defines an ordered set of strings: “first name,” “middle name,” and “last name” (BuildingSmart 2014; ISO 2004). Another important element in EXPRESS language is declaration. Instances of the above mentioned data types need to be defined through declarations. For example, “IfcAreaMeasure” is a data type declared in IFC2X3 schema which is an instance of the “real” data type. Declarations can be used to declare instances of data types, entities, subtype constraints, schemas, constants, functions, procedures, and rules.

IFC models take three main file formats: STEP file (SPF), eXtensible markup language (XML) and ZIP. IFC-SPF is the fundamental STEP file format with the file extension “.ifc.” It is the main file format used in exchanging BIM models and has been widely used in conducting BIM research (Yang and Eastman 2007; Lee 2009). IFC-XML is the file format with the file extension “.ifcXML.” It is intended for interoperability with XML tools (Teicholz 2013).

### **2.3.2 Previous Efforts in Extracting Information from Building Information Models**

Many efforts have focused on BIM information processing, especially information extraction from IFC models. Existing BIM information extraction efforts have taken various different approaches. For example, Kim et al. (2013) utilized ifcXML parsers (implemented in Ruby programming language) to extract spatial, quantity, material, and relational information of building elements from IFC-based BIMs, for automatically generating construction schedules. Zhang and Issa (2013) utilized an ontology (implemented in Java programming language) that was coded in web ontology language (OWL) to extract partial models of IFC-based BIMs based on the IFC schema, for reducing the size and complexity of BIMs. There are also existing efforts in extracting information from BIMs to support automated compliance checking. These efforts extract BIM information into different types of representations. For example, Yurchyshyna et al. (2008) and Pauwels et al. (2011) utilized an Extensible Stylesheet Language Transformations (XSLT) transformation method to extract information from an IFC-based BIM into a resource description framework (RDF) graph to support

regulatory requirement checking, in general. Sinha et al. (2013) utilized Revit Application Programming Interface (API) methods to extract building parametric data from Revit BIMs for supporting automated compliance checking against energy code criteria. Tan et al. (2010) utilized Java classes to extract wall attributes from IFC-based BIMs for supporting automated building envelope design checking against building code requirements. Further, the development of the ifcOWL ontology enables the extraction of IFC-based BIM information based on the domain knowledge captured in the ontology, which could further serve the purpose of compliance checking (Beetz et al. 2009; Kadolsky et al. 2014). In addition, commercial BIM software implementations such as ArchiCAD, Autodesk Revit, and Solibri Model Checker have their proprietary methods to access and extract information from IFC-based BIMs.

### **2.3.3 IFC Extension and Data Access for Extracting Information from IFC-Based Building Information Models**

For information extraction from IFC models to be successfully applicable to automated compliance checking tasks, the current IFC schema needs to be extended to capture the required concepts and relations for compliance checking purpose. Because of the goal of improving interoperability, the extension of an IFC schema is usually a set of coordinated efforts led by BuildingSmart (Amann et al. 2015; Lee and Kim 2011). However, this does not prevent an IFC schema to be extended upon needs within a specific organization or for a specific purpose. Researchers have proposed various ways to extend the IFC schema for ACC

purposes. For example, Tan et al. (2010) defined the Extended Building Information Model (EBIM) to incorporate building hygrothermal performance simulation results (from a simulation software) into an XML-language-represented IFC schema; Niemeijer et al. (2009) proposed to use abstract syntax trees of constraints to extend the IFC schema with missing concepts and relations; and the Singapore CORENET project extended the IFC schema using FORNAX (i.e., a C++ library to derive new data and generate extended views of IFC data) objects (Eastman et al. 2009). However, to avoid inconsistency and incompleteness in the extension of the IFCs, a more objective and automatic way of extending the IFC schema is needed.

For data access, the Java Standard Data Access Interface (JSDAI) is a standard data access interface (SDAI) application programming interface (API) for accessing and processing object-oriented data defined in EXPRESS-based data models. There are two types of data access methods in JSDAI: early binding and late binding. Early binding requires the availability of a specific EXPRESS model (i.e., the specific IFC schema) at the program compiling time, and accesses each entity and attribute in the known EXPRESS model with specialized access methods. For example, the attribute “OverallHeight” of an entity “IfcDoor” is accessed using the specialized method “getOverallHeight.” Late binding, on the other hand, does not require the availability of a specific EXPRESS model at the program compiling time, and accesses entities and attributes using generalized access methods. For example, the attribute “OverallHeight” of an entity “IfcDoor” is accessed using the generalized methods

“getExplicit\_attributes” and “get.” Late binding is more complex than the early binding; but in comparison to early binding, it is independent of specific EXPRESS models and is thus more flexible in data access and processing.

JSDAI could be used to support IFC-based BIM information access. Among the different techniques that could help access information in an IFC-based BIM such as Java Toolbox IFC2x3/IFC4 (IFC Tools Project 2013), ifcplusplus (ifcPlusPlus 2015), Open IFC tools (Open IFC Tools 2010), and IFCToolboX (Eurostep 2002), JSDAI stands out because it could access BIMs using their metadata at the EXPRESS model level, which makes it not limited to a certain version (or a limited number of versions) of IFC schema(s), as the other techniques are. The potential use of JSDAI in accessing IFC-based BIMs was recognized by a number of researchers (Vanlande et al. 2008; Isikdag et al. 2007; Steel et al. 2010). As a result, JSDAI was used in a few research efforts for accessing IFC-based BIMs, such as in Windisch et al. (2012), Cheng and Das (2013), and Grunewald et al. (2010). But the utilization of JSDAI in these studies still focused on a specific IFC parser generated using JSDAI (i.e., using early binding data access) based on a specific IFC schema (e.g., IFC2x3), which made their developed methods/tools limited to extraction based on the specific IFC schema version that was used. Late binding data access in JSDAI, on the other hand, could enable a more flexible and robust BIM information access (i.e., using any IFC schema version), which to the best of the author’s knowledge has not been explored in prior research studies yet.

## **2.4 Automated Reasoning**

### **2.4.1 Logic Reasoning**

With the advancement in computing technology, the field of automated reasoning also progressed quickly. A variety of approaches have been developed for reasoning purposes such as decision tree, rule-based, and case-based. Example techniques of automated reasoning include rewriting, unification, and search strategies. Different techniques have different properties and uses. For example, backtracking is a technique to try alternative logic clauses and goals (i.e., sub-problems to solve for solving the main problem) for unification with facts when unification of the current logic clause and goal with facts fails (Sterling and Shapiro 1986). Backtracking guaranties the discovery of all solutions to a problem at the end of the solving process. Discovering all solutions to a problem is important for many applications such as ACC (because of the need to discover all noncompliance instances). However, in applications where a single acceptable solution is sufficient, it would be a waste of time and resources to continue searching to find all solutions. So, the decision of using backtracking or not depends on the type of application. For search, breadth-first and depth-first are the two main strategies. Breadth-first search is more appropriate to use when there are infinite paths in the search space or solutions exist at shallow paths. On the other hand, when many solutions exist or all paths lead to a solution, depth-first search is more appropriate (Poole and Mackworth 2010). When solving problems using automated reasoning, it is critical to select from and adjust available techniques to obtain a balance between the

problem-solving efficiency and the quality/completeness of the solution.

A formally-defined logic could represent and reason about the complicated logic relations in construction regulations more efficiently than conventional programming if-then-else logic statements. However, formally-defined logic has not been widely-adopted in construction regulation rule representation. One reason is the heavy intelligent effort needed to develop a general framework that can accommodate different types of regulations. Adopting the conventional programming if-then-else logic statements for each type of regulation, on the other hand, is relatively straightforward. Logic is essential in many automated reasoning systems (Portoraro 2011). Different types of formal logics of varying degrees of descriptive capability have been developed, including: propositional logic, predicate logic [e.g., first order logic (FOL)], modal logic, and description logic (Zhu et al. 2011; Wang et al. 2008). Among the different types of formally-defined logics, FOL is most widely used for logic inference-making. FOL has more than one correct and complete proof calculi (i.e., cases where the derivable sequents are precisely the valid ones for the calculi), which makes FOL suitable for automated reasoning. FOL is based on first order language, which has been used mainly for deductive arguments since its creation. First order language was intended to “express conditions which things can satisfy or fail to satisfy” (Hodges 2001). The development of FOL-based frameworks and methods could be dated back to last century (Ryu and Lee 1995; Horton and Spencer 1997), and it is still evolving (Baumgartner and Suchanek 2006; Bos 2009; Krotzsch et al. 2015).

## 2.4.2 Horn Clauses and Logic Programming

Inference-making in FOL is most efficient using Horn Clause (HC) logic clauses. A HC is one of the most restricted forms of FOL. Inference-making in FOL is most efficient using HC logic clauses, because of such restricted form (Saint-Dizier 1994). A HC is composed of a disjunction of literals (predicates) of which at most one is positive. A predicate is the building block of a logic clause. A predicate consists of a predicate symbol and one or more arguments in parenthesis following the predicate symbol [e.g., the predicate “*wall(w)*” has one predicate symbol “*wall*” and one argument “*w*,” where “*w*” is a variable]. All HCs can be represented as rules that have one or more antecedents [i.e., left-hand sides (LHSs)] that are conjoined (i.e., combined using ‘and’ operator), and a single positive consequent [i.e., right-hand side (RHS)]. For example, “*thickness(t) ∧ exterior\_basement\_wall(w) ∧ has(w,t) ∧ greater\_than\_or\_equal(t, quantity(71/2, inches)) ⊃ compliant(t)*” is a HC; where “ $\wedge$ ” is the conjunctive operator (i.e., “ $A \wedge B$ ” means “A and B”) and “ $\supset$ ” is the implication operator (i.e., “ $A \supset B$ ” means “A implies B”). There are three types of HCs: (1) one or more antecedents and one consequent, (2) zero antecedents and one consequent, and (3) one or more antecedents and zero consequents. Inference-making using HCs could be automatically and efficiently conducted, which makes it suitable for supporting automated reasoning for ACC.

Logic programming is a widespread and important application of HCs (Portoraro 2011). Based on HC, logic programming can represent knowledge rules and facts in a ready-to-use



manner for automated reasoning. Prolog is the most widely-used logic programming language and reasoner. Prolog is declarative in contrast to other non-logical programming languages. For example, in typical procedural programming languages like C programming language, a programmer has to clearly define how to solve the problem step by step, whereas in Prolog, a programmer only needs to define how to represent the problem. The solution steps in Prolog are already defined by the built-in reasoner of Prolog through a set of organized automated reasoning techniques such as search strategies and backtracking.

### **2.4.3 Previous Automated Reasoning Efforts in the Construction Domain**

There have been a few efforts in utilizing automated reasoning in the construction domain, as early as 1980s (Alexander and Sidney 1987). Such efforts covered various tasks such as concrete structural design (Alexander and Sidney 1987), schedule review and generation (Dzeng et al. 2005; Chevallier and Russell 1998; Udaipurwala and Russell 2000), construction planning (Kartam et al. 1991), work space arrangement (Akinci et al. 2002), inspection planning (Gordon et al. 2008), construction simulation and visualization (Kataoka 2008; Loch-Dehbi and Plumer 2011; Weldu and Knapp 2012), litigation outcome prediction (Mahfouz and Kandil 2012), design team selection (Park and Koo 2011), collaborative design (Ugwu et al. 2002), structural damage assessment (Ross et al. 1990), and work condition compliance testing (Vries and Steins 2008). Many of these systems use rule-based or case-based reasoning. However, they do not have a semantic-based reasoning framework for the purpose of analyzing and checking the conformance of a design to applicable regulations.

They are limited in the level of knowledge representation and reasoning, if considered for use for the purpose of analyzing and checking the conformance of a design to applicable regulations.

## **2.5 Semantic Modeling and Information Processing**

### **2.5.1 Overview of Semantic Modeling and Ontologies**

In general, “semantics” studies the meanings of the words (Fritz 2006). A semantic model defines data/information entities and relationships between the entities (Hanis and Noller 2011). Ontology is a widely-used type of semantic model. The term ontology, meaning “the study of being or existence,” originated in philosophy. In the computer and the information science domains it refers to “an explicit specification of a conceptualization” (Gruber 1995). This definition establishes the features of an ontology: (1) an ontology is representing a conceptualization (i.e., an abstract, simplified view of a domain of interest); and (2) the representation of the conceptualization is explicit. An ontological model consists of concept hierarchies, relationships (between the concepts), and axioms (Noy and Hafner 1997). The axioms are used together with the concepts and relationships to define the semantic meaning of the conceptualization (El-Gohary and El-Diraby 2010). An ontology offers a computer-understandable, domain-specific representation of the knowledge in a domain of interest, in a reusable, extendable format.

Ontology could be utilized in any task involving information processing. A processing using

ontology is referred to as ontology-based processing or semantic processing. Semantic, ontology-based processing is expected to achieve higher performance in comparison to non-semantic processing, because domain knowledge (represented in an ontology) could help to identify or distinguish domain-specific terms and meanings (Saggion et al. 2007; Soysal et al. 2010).

Many efforts have been made in the area of ontology. For example, Juszczyszyn and Kołaczek (2009) proposed a framework for guiding the processes of ontology alignment and negotiation in a multi-agent environment; Kołaczek and Juszczyszyn (2010) proposed a general framework for decision-making about ontology alignment and negotiation which takes into account the properties of the actual communication network and utilizes the deontic logic formalism for reasoning; Maynard et al. (2006) discussed existing evaluation metrics, and proposed a new one for evaluating the ontology population task; Rubino et al. (2004) presented a web ontology language (OWL) ontology of fundamental legal concepts developed within the European project for Standardized Transparent Representations in order to Extend Legal Accessibility (ESTRELLA); Gruninger and Fox (1995) described a methodology for guiding the design of ontologies, as well as providing a framework for evaluating the adequacy of these ontologies; Malik et al. (2010) discussed knowledge management and semantic annotation and presented a framework for it using the General Architecture for Text Engineering (GATE) and ontology. The framework is intended for intelligent information retrieval.

In the construction domain, there have been several research efforts for developing or utilizing ontologies. For example, El-Diraby and Zhang (2006) presented a taxonomy for building construction. It is the first attempt to present building construction knowledge in a semantic way; Beetz et al. (2009) described an effort to semiautomatically transform an IFC model in EXPRESS format into an OWL-represented ontology; El-Diraby and Kashif (2005) presented a distributed ontology architecture for knowledge management in highway construction as an extension for the e-COGNOS ontology; Anumba et al. (2008) reviewed the fundamental concepts and roles of ontologies in the construction project delivery process; and El-Gohary and El-Diraby (2010) proposed a domain ontology for infrastructure and construction processes.

### **2.5.2 Ontology Development and Evaluation**

Ontology development could be conducted by a number of methods such as the Toronto Virtual Enterprise (TOVE) method, Enterprise model method, METHONTOLOGY method, and IDEF5 method (Jones et al. 1998; Corcho et al. in 2003). The TOVE is a first-order logic approach to develop ontology elements in six steps: (1) motivating scenarios, (2) informal competency questions, (3) terminology specification, (4) formal competency questions, (5) axiom specification, and (6) completeness theorems (Gruninger and Fox 1995). Enterprise model method is a skeletal methodology to develop an ontology in four steps (Uschold 1995). The METHONTOLOGY is a comprehensive ontology development methodology that builds an ontology from scratch (or reusing other ontologies) using seven steps: (1) specification, (2)

knowledge acquisition, (3) conceptualization, (4) integration, (5) implementation, (6) evaluation, and (7) documentation (Fernandez et al. 1997). The IDEF5 method is a general ontology development procedure with five main guidelines: (1) organizing and scoping, (2) data collection, (3) data analysis, (4) initial ontology development, and (5) ontology refinement and validation (KBSI 1994). Existing methods could be classified into task-oriented methods and comprehensive development methods (Jones et al. 1998; Vrandečić 2010). A task-oriented method takes a task as a starting point and focuses on the task as the functions of the ontology. A comprehensive development method takes a stage-based approach to develop an ontology through well-defined stages or takes an evolving prototype approach to iteratively refine a prototype ontology (Jones et al. 1998).

An ontology could be evaluated using a number of different evaluation methods, and these methods were summarized into four general categories: (1) evaluation methods based on comparing the ontology to a gold standard; (2) evaluation methods based on the application results; (3) evaluation methods based on comparing the ontology with a source of data; and (4) evaluation methods based on expert assessment using a set of predefined criteria, standards, and requirements (Brank et al. 2005).

### **2.5.3 Semantic Modeling and Semantic NLP**

A semantic model aims at capturing the meanings (thus knowledge) of a domain or topic, usually in a structured manner. Ontology is a widely-used type of semantic model that

captures domain knowledge in the form of concept hierarchies, relationships between concepts, and axioms. The axioms are used together with the concepts and relationships to define the semantic meaning of the conceptualization. An ontology is easily reusable and extendable (El-Gohary and El-Diraby 2010). The use of a semantic model could help in NLP tasks. For example, semantic-based information extraction has been shown to achieve better performance than syntactic-only information extraction (Zhang and El-Gohary 2011; Soysal et al. 2010) and text classification (Zhou and El-Gohary 2014).

WordNet is a slightly different semantic model than an ontology, which was also frequently utilized in semantic research efforts. It is a large lexical database of English where the four types of POS words (nouns, verbs, adjectives, and adverbs) are grouped into sets of cognitive synonyms (synsets) (Fellbaum 2005). In WordNet, each of the four POS categories is organized into a subnet and the synsets are linked to each other using one or more of the following conceptual semantic and lexical relations: synonymy, hyponymy (sub-super or is-a relation), meronymy (part-whole relation), and antonymy (Fellbaum 2005). Because of the abundant, explicitly-defined and well-structured conceptual semantic relations between word senses in WordNet, WordNet has been widely used in semantic NLP research, as a “lexical database” (Shehata 2009; Kamps et al. 2004), a “lexical dictionary” (Varelas et al. 2005), a “semantic dictionary” (Simpson and Dao 2010), or a “domain-independent background knowledge model” (Suchanek et al. 2007). The lexical relations in WordNet can assist in semantic text processing. The hyponymy and meronymy relations in WordNet correspond

well to the is-a and part-whole relations in semantic models. In addition, a synonymy relation carries an “equivalency” relation between semantic classes.

Although semantic relations are generally domain-dependent (Orna-Montesinos 2010), WordNet is widely used for domain-specific text processing tasks. This could be attributed to two main reasons: (1) a lack of domain-specific lexical/relation databases with coverage comparable to that of WordNet. For example, the most relevant lexical/relation database effort in the building domain, the International Framework for Dictionaries (IFD) by buildingSMART, is still being tested and is currently not openly accessible; and (2) despite being general (as opposed to domain-specific), the dictionary-level coverage in WordNet could be useful in helping identify basic semantic relations between words or concepts. A few research efforts in the AEC domain [e.g., Orna-Montesinos (2010) and Li (2010)] have utilized WordNet in text/knowledge processing or analysis.

Semantic Similarity (SS) is the conceptual/meaning distance between two entities such as concepts, words, or documents (Slimani 2013). SS plays an important role in information and knowledge processing tasks such as information retrieval (Rodri'guez and Egenhofer 2003), text clustering (Song et al. 2014), and ontology alignment (Jiang et al. 2014).

SS could be quantitatively estimated using different measures. Some popular measures are: (1) Shortest Path Similarity, which utilizes the shortest path connecting two concepts in a taxonomy (i.e., concept hierarchy) (Resnik 1995); (2) Leacock-Chodorow Similarity, which

utilizes the shortest path connecting two concepts in a taxonomy while penalizing long shortest path according to the depth of the taxonomy (Resnik 1995); (3) Resnik Similarity, which utilizes the information content measure from information theory to measure the information shared by two concepts using the information content of the two concepts' least common subsumer (Resnik 1995); (4) Jiang-Conrath Similarity, which utilizes the information content of the two concepts in addition to that of their least common subsumer in the taxonomy; and (5) Lin Similarity, which utilizes the ratio between the information content of the least common subsumer (in the taxonomy) of the two concepts and the sum of the information contents of the two concepts.

Shortest Path Similarity is simple and intuitive; it approximates the conceptual distance between concepts by the number of edges in-between. The main limitation of Shortest Path Similarity is its inability to take specificity of concepts into consideration, which leads to same similarity results for a concept pair at a shallow taxonomical level and another concept pair at a deep taxonomical level as long as the counts of number of edges for both concept pairs are the same. This limitation is compensated for in Leacock-Chodorow Similarity by taking the maximum depth of the taxonomy into consideration. Thus, using Leacock-Chodorow Similarity, if two concept pairs have an equal number of edges, the taxonomically deeper pair (i.e., more specific) would have a larger similarity score than the taxonomically shallower pair. Resnik Similarity, Jiang-Conrath Similarity, and Lin Similarity are information content-based similarity measures. Resnik Similarity is sometimes considered



insufficiently-discriminative because different concept pairs could have the same least common subsumer. Jiang-Conrath Similarity and Lin Similarity improve upon Resnik Similarity by taking the information content of the two concepts into consideration, in addition to their least common subsumer. The main limitation of information content-based measures, however, is the need of using a text corpus in addition to the taxonomy for similarity assessment, which may lead to variability in similarity results depending on the corpus that is used.

When using the above-mentioned measures to evaluate SS between words, WordNet is commonly used as the taxonomy. However, when using WordNet, these measures assess SS between terms, but not concepts which could be multi-term.

In addition to ontology and WordNet, outputs from certain NLP tasks could be used in semantic analysis of text, such as semantic relation output from semantic parsing and named entity output from named entity recognition. Semantic parsing is the task of converting text into a formal meaning representation (Clarke et al. 2010). Various methods have been developed for semantic parsing (Cai and Yates 2013; Clarke et al. 2010; Farkas et al. 2010). However, these development mostly focused on a general domain (in comparison to domain specific development) and were thus still limited in performance. The best F1-measures were around 80-87% (Lu 2014; Farkas et al. 2010). Named entity recognition is the task of recognizing information units from text such as names (including person, organization and location names) and numeric expressions (including time, date, money and percent

expressions) (Nadeau and Sekine 2007). Various methods have been developed for named entity recognition (Nadeau and Sekine 2007) and high performance (i.e., higher than 90% F-measure) were achieved for certain categories such as person names (Chieu and Ng 2003; Nothman et al. 2012). However, existing named entity recognition methods are still limited; they can only recognize a limited set of specific categories (e.g., organization, location) and thus cannot be directly used for recognizing all named entities in a domain (Nadeau and Sekine 2007).

#### **2.5.4 Semantic Modeling and Building Information Modeling**

Semantic modeling is important to building information modeling in two ways: (1) Because of the object-oriented nature of BIMs, a data schema of BIMs must be a semantic model; and (2) semantic models are utilized to help process information in BIMs.

Data schemas of BIMs are semantic models. As the most popular and ISO-registered BIM data schema, the IFC schema provides a common data schema across BIMs. The IFC schema was considered having a longer history and more development effort than most other existing schemas in the AEC industry (Torma 2015). In addition to IFC, different BIMs schema have been developed for different purposes. For example, the Green Building XML (gbXML) data schema was developed for green building design, the City Geography Markup Language (CityGML) was developed for presenting 3D objects and their visible surfaces in urban environments (Anjomshoaa et al. 2015). Because of the need of interoperability among BIMs,

research efforts have been conducted to align these data schemas with each other in different ways. For example, Delgado et al. (2013) tried to align a CityGML ontology and an IFC ontology using 15 ontology matching techniques. Mignard and Nicolle (2014) developed a semantic extension to BIM in the ACTIVE3D platform for building an extensible ontology that can be instantiated by information based on both the IFC ontology and the CityGML ontology.

Semantic models are used to support information processing in BIMs by providing domain-specific knowledge. Similar to the semantic models used in helping with NLP tasks, the semantic models used in helping with BIM information processing are mostly ontologies. For example, Lee et al. (2014) proposed an ontology to support search automation in BIMs for building elements and materials to facilitate cost estimation using BIMs. Costa and Madrazo (2015) used ontologies to link building component catalogues with BIMs to facilitate building product information service using BIMs. Lee and Jeong (2012) utilized ontology-based filters to translate design data in BIMs into domain-specific (e.g., architectural, structural, or mechanical) data to facilitate shared understanding among designers in different disciplines on the same BIM-based design. Thus, semantic models not only lay the foundation for BIMs, but also keep increasing its information processing capabilities to support more applications in better ways.

### **2.5.5 Semantic Modeling and Automated Reasoning**

A semantic model offers a meaning-rich representation of the knowledge of a domain that is formal and computer-processable. Such representation facilitates enhanced reasoning through leveraging the captured domain knowledge. Because ontologies allow for high level information and knowledge representation and are key to enable complex automated reasoning, the majority of research and development efforts in the area of automated reasoning rely on the use of ontologies (Baumgartner and Suchanek 2006; Stenmark and Malec 2014; Ivanovic and Budimac 2014). For example, in the biomedical domain, ontologies can be used for automated reasoning on knowledge contained in Clinical Practice Guidelines and Care Pathways (Ivanovic and Budimac 2014). In the safety engineering domain, ontologies can be used for automated reasoning about the location information of past accidents (Batres et al. 2014). In the education domain, ontologies can be used for automated reasoning in generating multiple choice questions (Al-Yahya 2014).

### **2.6 Machine Learning Algorithms**

Machine learning (ML) is a discipline that studies algorithms that can learn from data (Kovahi and Provost 1998). The algorithms usually appear as models that take inputs and make predictions based on the inputs (Bishop 2006). In any ML application, different ML algorithms are usually tried out and tested. Some of the most commonly-used ML algorithms are summarized in Table 2.2.

Table 2.2 Commonly-used Machine Learning Algorithms

	Machine learning algorithm				
	Naïve Bayes	Perceptron	Decision Tree	k-NN	SVM
Key feature	simple but effective	linear	flexible	similarity-based	kernel-based

Naïve Bayes is a simple statistical ML algorithm. It applies Bayes' rule to compute conditional probabilities of predictions given evidence. It is the simplest type of algorithm among the commonly-used ML algorithms. However, Naïve Bayes could outperform more complex learning algorithms in some cases (Domingos 2012).

Perceptron is a linear learning algorithm where predictions are made based on a linear combination of feature vectors (Rosenblatt 1958). Perceptron is applicable to problems that are linearly separable. The application process of perceptron is iterative: a prediction vector is iteratively constructed based on each instance in the training dataset (Freund and Schapire 1999).

Decision tree is a ML algorithm that uses a tree to map instances into predictions. In a decision tree model, each non-leaf node represents one feature, each branch of the tree represents a different value for a feature, and each leaf node represents a class of prediction. Decision tree is a flexible algorithm that could grow with increased amount of training data (Domingos 2012).

K-Nearest Neighbor (k-NN) is a similarity-based ML algorithm. K-NN predicts the class of an instance using the instance's k nearest instances by assigning it the majority class of those k instances' classes (Cover 1967; Domingos 2012). Depending on the task, k values in

different ranges need to be tested to find the best-performing k-NN classifier (Gunavathi and Premalatha 2014). K-NN is sensitive to noise (Gunavathi and Premalatha 2014). It performs better when utilized in training classifiers using small datasets (Raikwal and Saxena 2012). K-NN is typically outperformed by Support Vector Machines (SVM) (with small differences), but occasionally could achieve equally or slightly better performance than SVM (Sudha and Bhavani 2012) when the influence of the nonlinear relationship between the features and a class is not dominating the performance and neither noises nor unbalanced samples in the training data are dominating the performance.

Support Vector Machines (SVM) is a kernel-based ML algorithm that has significant computational advantages over standard statistical algorithms. A kernel method is a technique for constructing nonlinear features so that nonlinear functional relationships could be represented using a linear model. A linear model is much simpler comparing to a nonlinear model, both theoretically and practically, giving SVM its computational advantages (Cristianini and Shawe-Taylor 2000). Gaussian kernel and polynomial kernel are two commonly-used kernels (Hofmann 2006). SVM was found to outperform other ML algorithms in many applications such as text classification (e.g., Salama and El-Gohary 2013a), although in certain cases other algorithms (such as k-NN) outperformed SVM (Vo et al. 2015). The infrequent cases where SVM was outperformed by other algorithms are usually for tasks and/or data where the influence of the nonlinear relationship between the features and a class is not dominating the performance, such as in the case of using spatial

and wavelet type of features in classifying human gait patterns in Sudha and Bhavani (2012).

ML is one type of machine-based reasoning (i.e., inductive reasoning), where the various types of ML algorithms induct knowledge from input data (Domingos 2012). In any machine-based reasoning, successful reasoning depends on appropriate representations (Bundy 2013). What features should be used to represent the data in a ML problem is, thus, an important decision.

### **3 CHAPTER 3 – AUTOMATED INFORMATION EXTRACTION FROM BUILDING CODES**

#### **3.1 Comparison to the State of the Art**

Many research efforts were conducted for information extraction in various domains (Soysal et al. 2010; Sapkota et al. 2012; Hogenboom et al. 2013). The state-of-the-art semantic information extraction studies have four major focuses: named entity extraction, attribute extraction, relation extraction, and event extraction. Named entity extraction, attribute extraction, and relation extraction aim to extract instances of a single concept (e.g., named entity) or of two related concepts (Ling and Weld 2012; Pasca 2011; Wang et al. 2010). Event extraction aims to extract instances of multiple concepts (Patwardhan 2010). From this perspective, the proposed approach is more similar to event extraction because instances of multiple concepts in a provisional requirement are extracted. However, compared with event extraction, the approach is different in two primary ways. First, the information is extracted in a more flexible manner. In the proposed approach, two types of information elements are defined: “rigid information elements” and “flexible information elements.” A rigid information element has a predefined, fixed number of concepts/relations (e.g., in a terrorist event case, it is predefined that “victim” is associated with only one concept). In contrast, a flexible information element has a varying number of concepts/relations depending on the instance at hand (e.g., in this approach, “subject restriction” has a varying number of multiple concepts/relations). Unlike event extraction, the proposed approach can extract the instances



of flexible information elements. Second, because a method for extracting information elements in a more flexible way is introduced, a deeper level of information extraction is performed (i.e., a deeper level toward full sentence interpretation). Shallow NLP conducts partial analysis of a sentence or analyzes a sentence from a specific angle of view (e.g., part-of-speech tagging, text chunking). Deep NLP aims at full sentence analysis, with a more complex understanding of the text toward capturing the entire meaning of sentences (Zouaq 2011). Correspondingly, shallow information extraction extracts specific type(s) of information from a sentence, while deep information extraction aims at extracting all information that is expressed by a sentence based on the full analysis of the sentence.

In terms of information extraction performance, for the four main types of information (entities, attributes, relations, and events), state-of-the-art performance results are within the range of 80% to 90% for both precision and recall (e.g., Li et al. 2012; Bing et al. 2013; Sun et al. 2011; Tang et al. 2012). One of the recent information extraction studies that aimed to extract protected health information reported a best performance of 96.68% and 93.77% for precision and recall, respectively (Deleger et al. 2013).

In the construction domain, a number of important research efforts utilized NLP techniques (e.g., Caldas and Soibelman (2003) conducted ML-based text classification of construction documents); however, only a few of these efforts conducted some type/level of information extraction (e.g., Abuzir and Abuzir (2002) and Al Qady and Kandil (2010)). Al Qady and Kandil (2010) used shallow parsers to extract concepts and relations from construction

contracts. In Al Qady and Kandil (2010), (1) the extraction is only based on syntactic features produced by shallow parsing; and (2) information recognition is based on specific types of phrases and their roles (produced by shallow parsing) [e.g., NP segment and its role SUBJ (i.e., subject)], which allows for extracting relations between concepts. In the proposed approach, (1) semantic features are used in addition to syntactic ones; and (2) patterns that consist of a variety of syntactic and semantic features are used in information extraction and conflict resolution rules, which allows for a deeper level of information extraction (i.e., extracting all information of a requirement for further representation in a logic-based rule format). Abuzir and Abuzir (2002) used information extraction techniques to extract terms and relations from HyperText Markup Language (HTML) documents for constructing a civil engineering thesaurus. In Abuzir and Abuzir (2002), (1) the extraction uses HTML-based document structure features (including title tags, heading tags, and URLs) and simple lexical syntactic features; and (2) because the main purpose of the extraction is thesaurus construction, their information extraction focuses on extracting terms. In the proposed approach, (1) document structure features are not used (because of dealing with unstructured text rather than HTML documents) and the extraction relies on the syntactic and semantic features of the text; and (2) because the ultimate purpose is automated reasoning about regulatory requirements, information extraction is conducted on a deeper level; not only terms/concepts need to be extracted, but also other information elements (e.g., restrictions) need to be extracted for extracting all information expressed in a sentence/requirement. As

such, compared with these efforts, this research (1) addresses a different application (i.e., ACC). NLP methods, algorithms, and results are highly application-dependent (Salama and El-Gohary 2013a); (2) tackles a deeper NLP/information extraction task. This research aims to automatically process the text to extract regulatory requirements/rules and represent them as logic clauses; and (3) taking a deeper semantic approach for NLP. In this research, a domain ontology for identifying semantic text features is utilized. Using domain-specific semantics and ‘flexible information elements’ to achieve relatively deep semantic NLP allows for: (a) analyzing complex sentences that would otherwise be too complex for automated information extraction, (b) recognizing domain-specific text meaning, and (c) in turn, improving performance of information extraction.

### **3.2 Proposed Information Extraction Method and Algorithm**

This section presents the proposed method for automatically extracting information from building codes (.txt format, excluding both tables and figures). The method is presented as a domain-specific, semantic information extraction method that can be adopted (as is or with adaptation) by other researchers in the construction domain. The method is composed of the following seven phases (as per Figure 3.1): information representation, preprocessing, feature generation, target information analysis, development of information extraction rules [information extraction (IE) and conflict resolution (CR) rules], extraction execution, and evaluation. The approach is iterative to improve performance.

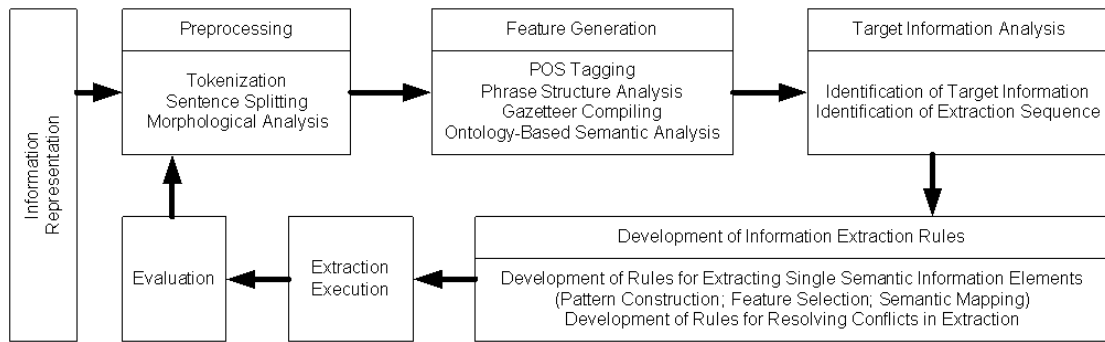


Figure 3.1 Proposed Information Extraction Method

### 3.2.1 Phase I – Information Representation

This phase is proposed to define the representation format for the extracted information. In this method, the ultimate representation format is one or more logic rules that could be directly used for automated compliance reasoning. For intermediate processing, a new ACC-tuple is proposed to represent the extracted information. The use of a tuple format for intermediate processing is proposed because it is easy for computer manipulation and evaluation (e.g., <Subject, Attribute, Value> is a three-tuple).

In the ACC-tuple representation, each element is called a “semantic information element,” which is: (1) an ontology concept; (2) an ontology relation; (3) a “deontic operator indicator,” which is a term indicating an obligation, permission, or prohibition following the semantic ACC model in Salama and El-Gohary (2013b); or (4) a “restriction,” which is an element that places a constraint on the definition of another semantic information element, where the constraint is expressed in terms of ontology concepts and relations. The following types of semantic information elements are introduced: a ‘simple semantic information element’ (SIE)

versus a ‘complex SIE,’ and a ‘rigid SIE’ versus a ‘flexible SIE.’ A simple SIE is associated with a single concept/relation/indicator, whereas a complex SIE is expressed in terms of multiple concepts and relations. The simple SIEs are rigid, whereas the complex SIEs are flexible. A rigid SIE is an information element with a predefined, fixed number of concepts/relations, whereas a flexible SIE has a varying number of concepts/relations depending on the instance at hand. Accordingly, in the ACC-tuple, an ontology concept, an ontology relation, and a deontic operator indicator are simple (and thus rigid) SIEs, whereas a restriction is a complex (and thus flexible) SIE. The use of flexible SIEs is key to providing the flexibility needed to facilitate full sentence analysis. A specific word, phrase, or chunk of text extracted and mapped according to a SIE is referred to as an “information element instance.”

To prepare for further information transformation into logic rules, a semantic mapping step is used to match the extracted information element instances to their respective semantic concepts: (1) for ontology concepts and relations, their information element instances are mapped to the corresponding concepts and relations. For example, “courts” is mapped to “court,” “net area” is mapped to “net\_area,” “not less than” is mapped to “greater\_than\_or\_equal;” (2) for deontic operator indicators, their instances are mapped to the indicated deontic concepts. For example, “shall” is mapped to “obligation;” and (3) for restrictions, their instances are decomposed and mapped to one or more ontology concepts and relations. For example, “between the insulation and the roof sheathing” is mapped to

“relation(between, insulation, roof\_sheathing) .”

The extracted information element instances (in ACC-tuple format) – after conducting necessary semantic mapping – are further transformed to HC-type logic rules (as shown in Table 3.1) for logic-based deduction and reasoning about compliance. The method/algorithm for information transformation is presented in Chapter 4.

Table 3.1 Example of Extracted Semantic Information Elements and Their Corresponding Logic Representation

Information tuple extracted from text sentences	Subject	airspace
	Subject restriction	relation (between, insulation, roof_sheathing)
	Compliance checking attribute	NA
	Deontic operator indicator	obligation
	Quantitative relation	provide
	Comparative relation	greater_than_or_equal
	Quantity value	1
	Quantity unit/reference	inch
	Quantity restriction	NA
Horn clause logic representation		$\forall (a, i, r, s) ((\text{airspace}(a) \wedge \text{insulation}(i) \wedge \text{roof\_sheathing}(r) \wedge \text{between}(a, i, r) \wedge \text{has}(a, s)) \supset O(\text{greater\_than\_or\_equal}(s, \text{quantity}(1, \text{inch})))$

Note: Universal quantifier (‘ $\forall$ ’ or ‘for all’) asserts that the sentence is true for all instances of a variable; Conjunction ‘ $\wedge$ ’: ‘ $A \wedge B$ ’ indicates that ‘A’ is true and ‘B’ is true; Implication ‘ $\supset$ ’: ‘ $A \supset B$ ’ indicates that ‘A’ implies ‘B’ (if ‘A’ is true then ‘B’ is true); Obligation operator (O):  $O A$  indicates that ‘A’ is obligated.

### 3.2.2 Phase II – Preprocessing

This phase is used to prepare the raw (i.e., unprocessed) text for further processing. In the proposed method, preprocessing consists of tokenization, sentence splitting, dehyphenation, and morphological analysis.

### 3.2.2.1 Tokenization

Tokenization is the process of dividing the sequences of characters (pure strings) in the text into units (sentences or words) (Grefenstette and Tapanainen 1994). This process aims to prepare the text for further unit-based processing, such as sentence splitting and POS tagging, and is conducted based on parsing the text according to common delimiters (i.e., white spaces and punctuations) with disambiguation consideration (e.g., “,” as a delimiter in a number instead of punctuation). In the proposed method, tokenization divides the sequences of characters into tokens, where a token is a single word, a number, a punctuation mark, a white space, or a symbol (e.g., “&,” and “\$”). For example, as shown in Figure 3.2, each word, number, and punctuation mark was recognized and labeled as a token.

### 3.2.2.2 Sentence Splitting

Sentence splitting is the process of recognizing each sentence of the text. Similar to tokenization, sentences are recognized based on typical sentence boundaries (i.e., periods, exclamation marks, and question marks) with disambiguation consideration (e.g., recognizing “.” as a decimal point in a number instead of a period). In the proposed method, the result of sentence splitting is a set of sentence segmentations (with recognized boundaries). For example, as shown in Figure 3.2, the boundaries of the sentence were recognized and labeled out using the “<sentence>” (i.e., starting of a sentence) or “</sentence>” (i.e., ending of a sentence) tags.

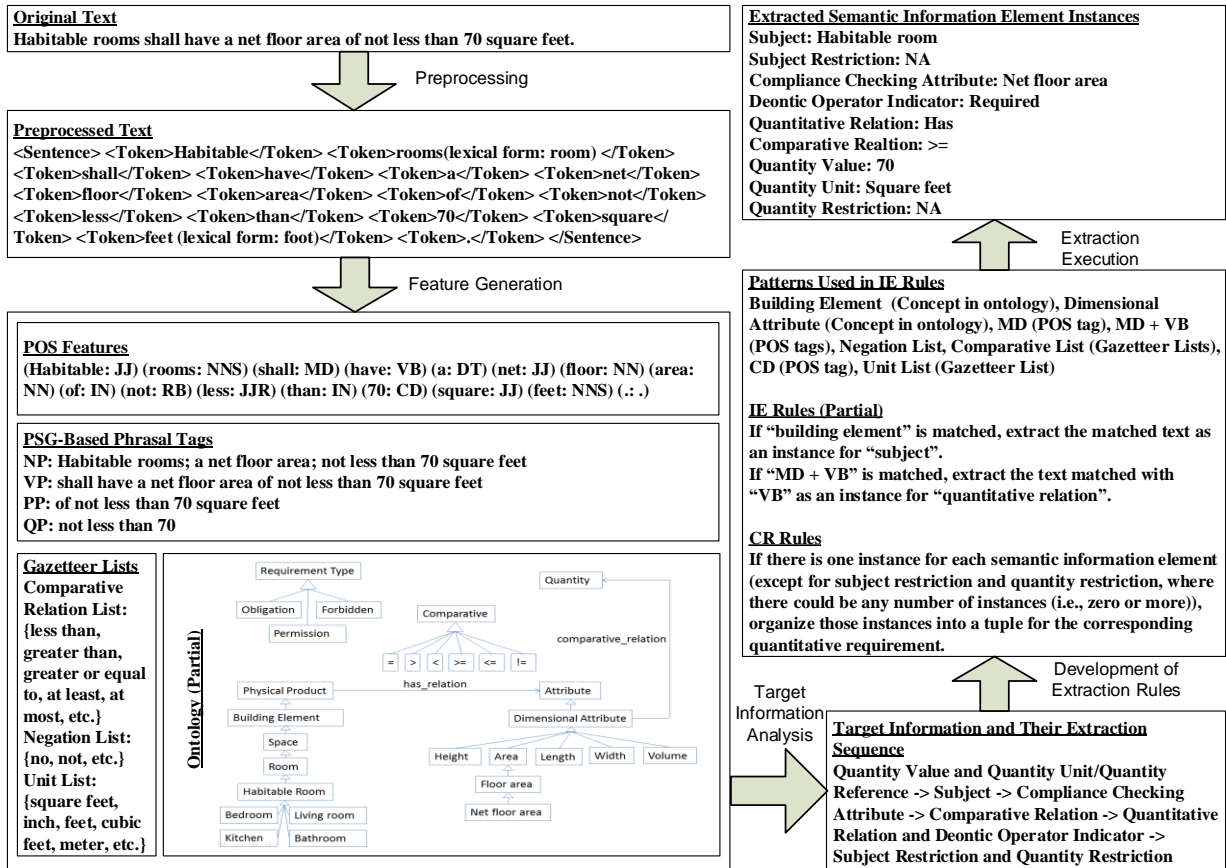


Figure 3.2 Illustrative Example Applying Proposed Information Extraction Method

### 3.2.2.3 Morphological Analysis

Morphology refers to the study of composition and structure of words. Morphological analysis (MA) aims to recognize the different forms of a word and to map them to the lexical form of that word in a dictionary (Fautsch and Savoy 2009). MA maps various nonstandard forms of a word (e.g., plural form of noun, past tense of verb) to its lexical form (e.g., singular form of noun, infinitive form of verb). For example, “constructs,” “constructed,” and “constructing” are all mapped to “construct.” Additionally, as shown in Figure 3.2, “rooms” and “feet” were mapped to their lexical forms “room” and “foot,” respectively. Whereas tokenization and sentence splitting are essential for information extraction because the text



must be broken down into units for further processing, MA is not essential for information extraction but is used to improve the identification of words with the same lexical form. The proposed preprocessing methodology incorporates MA because it aids in the recognition of ontology concepts. For example, the plural form of a concept could be recognized although the ontology uses only the singular form.

#### 3.2.2.4 Dehyphenation

Dehyphenation is used to remove hyphens that indicate continuations of words across two lines. Doing so prevents a word from not being recognized because of such a hyphen.

### **3.2.3 Phase III – Feature Generation**

This phase generates a set of features that describe the text. The proposed method uses domain-specific ontology-based semantic features, in addition to syntactic features and proposes the use of PSG-based phrasal tags to reduce the number of needed patterns. The proposed feature generation methodology consists of POS tagging, phrase structure analysis (using PSG), gazetteer compiling, and ontology-based semantic analysis. Syntactic features, such as POS tags, are widely used for information extraction, as in Afrin (2001). Semantic features benefit information extraction tasks beyond solely using syntactic features because they express domain-specific meaning/knowledge, as in Soysal et al. (2010). In the proposed method, both syntactic (POS tags, PSG-based phrasal tags, gazetteer terms) and semantic features (concepts and relations) are generated; subsequently, these features are used to

define patterns (text patterns in the proposed IE and CR rules that aid in the process of pattern matching for information extraction).

### 3.2.3.1 Part-of-Speech Tagging

POS tags are the labels assigned to words of a sentence that indicate their lexical and functional categories showing the structure inherent in the language. POS tagging aims to tag each word with the POS of the word, such as NN (singular nouns), JJ (adjectives), VB (verb), CC (coordinating conjunctions) (Galasso 2002). For example, as shown in Figure 3.2, “floor,” “Habitable,” and “have” were tagged as NN, JJ, and VB, respectively. In the proposed method, the POS tagging process also tags other tokens, such as numbers, punctuations, and symbols.

### 3.2.3.2 Phrase Structure Analysis

The proposed phrase structural analysis builds on the POS tagging step and aims to assign type labels (phrasal tags) to phrases of a sentence. Examples of phrasal tags are NP (noun phrase), VP (verb phrase), and PP (prepositional phrase). For example, as shown in Figure 3.2, “Habitable rooms,” “shall have a net floor area of not less than 70 square feet,” and “of not less than 70 square feet” were assigned NP, VP, and PP tags, respectively. In the method, PSG is used to generate phrasal tags. Application-specific PSG rules are derived based on a randomly selected sample of text (called, here, “development text,” which is also used for text analysis and further development of IE and CR rules). Applying these PSG rules, phrasal tags are assigned when a certain combination of POS tags and/or phrasal tags are encountered.

For example, the rule “QP → JJR IN CD” states that the phrasal tag “QP” (quantifier phrase) should be assigned when the sequence of POS tags “JJR IN CD” is encountered, as in the phrase, “less (JJR) than (IN) 0.07 (CD).” The use of phrasal tags together with PSG reduces the possible number of enumerations in patterns. For example, the three PSG rules NP → NP PP; NP → DT NN; and PP → IN NP together enable the phrasal tag feature NP to match many (actually infinite number of) noun phrases expressed by recursively attaching prepositional phrases to a base noun, such as “the wall,” “the wall of the room,” “the wall of the room in the building,” “the wall of the room in the building with a vent,” “the wall of the room in the building with a vent at the bottom.” In this step, PSG is derived from previously POS-tagged source text and is subsequently used to assign PSG-based phrasal tags to sentences in the source text.

To empirically study the effect of utilizing PSG-based phrasal tags on the number of patterns, an experimental test was conducted for preliminary verification of the proposed method. The author developed the patterns for extracting “subjects” two times: one time with PSG-based phrasal tags and one time without. Twenty-two (22) and 46 patterns were needed, with and without PSG-based phrasal tags, respectively, indicating that the use of PSG-based phrasal tags in pattern construction reduces the number of needed patterns in IE rules.

### 3.2.3.3 Gazetteer Compiling

A gazetteer is a set of lists containing names of specific entities (e.g., cities, organizations) (Cunningham et al. 2011). In general, a gazetteer list groups any set of terms based on any

specific commonality possessed by these terms. In the proposed method, the information that a word or phrase belongs to a certain list in the gazetteer is used as a feature for information extraction tasks. Different gazetteer lists are available [e.g., lists for currency, data units, and cities in the A Nearly-New IE System (ANNIE) Gazetteer of the General Architecture for Text Engineering (GATE) platform]. The use of a gazetteer in automated information extraction aids in recognizing terms based on those commonalities (Maynard et al. 2004). In the proposed method, a gazetteer is used to provide a set of term lists, in which each list has a specific function. For example, terms such as “no” and “not” have a function “negation,” and, as such, are included in the author’s “negation gazetteer list.” In the proposed method, several types of gazetteer lists are compiled and used, such as the “comparative relation gazetteer list,” which is composed of terms indicating comparative relations, including “greater than or equal,” “less than or equal,” “at most,” and “at least.” For example, as shown in Figure 3.2, “not,” “less than,” and “square feet” were in the “negation gazetteer list,” “comparative relation gazetteer list,” and “unit gazetteer list,” respectively. The information presented in a gazetteer list could have been represented as part of an instantiated ontology (e.g., the list of countries could have been represented as instances of the concept “country”). However, for computational efficiency, such instances were separated from the ontology (in the form of gazetteer lists).

#### 3.2.3.4 Ontology-Based Semantic Analysis

Ontologies are used to represent domain knowledge. A construction domain ontology offers a

semantic representation of the knowledge in the construction domain and, thus, could aid in extracting relevant information based on domain-specific meaning. In the proposed method, the concepts and relations of an ontology help extract the semantic features of the text and, thus, aid in semantic information extraction. Figure 3.2 shows a partial (and schematic) view of the used ontology, including its concepts (e.g., dimensional attribute) and subconcepts (e.g., floor area).

To verify the selection of a semantic approach, by comparing the results of semantic information extraction to that of syntactic-only information extraction, an experiment on extracting quantitative requirements from a randomly selected section of Chapter 12 of IBC 2006 – Section 1203 – was conducted. Table 3.2 shows the comparative results in terms of precision, recall, and F1-measure. The results show that semantic information extraction outperforms syntactic-only information extraction, with an increase in precision from 85% to 96% and an increase in recall from 81% to 92%.

Table 3.2 Comparative Testing of Syntactic-Only Information Extraction and Semantic Information Extraction: Experimental Results for Section 1203 of Chapter 12 of IBC 2006

Performance measure	Syntactic-only IE	Semantic IE
Precision	85%	96%
Recall	81%	92%
F1-measure	83%	94%

### 3.2.4 Phase IV – Target Information Analysis

This phase is proposed to manually analyze the text to identify the types of semantic information elements to be extracted and their interrelationships, and the sequence of their

extraction. In the proposed method, an approach for separation and sequencing of semantic information elements (SSSIE) is proposed to reduce the number of needed information extraction patterns.

#### 3.2.4.1 Identification of Target Information

In this step of the method, the development text is manually analyzed to identify the types of requirements that are expressed in the text (e.g., quantitative requirement). Based on domain knowledge (expressed in the ontology), the types of semantic information elements that are needed to represent the types of requirements are defined. For example, if the information to be extracted is related to terrorist attack events, then the types of semantic information elements could include “perpetrator individual,” “perpetrator organization,” “target,” “victim,” and “weapon.” For the example in Figure 3.2, the information to be extracted is related to quantitative requirements. So the following types of semantic information elements were identified: “subject,” “compliance checking attribute,” “deontic operator indicator,” “quantitative relation,” “comparative relation,” “quantity value,” “quantity unit,” “quantity reference,” “subject restriction,” and “quantity restriction.”

#### 3.2.4.2 Identification of Extraction Sequence

This step identifies the sequence of extracting the semantic information elements. The experimental studies of this research showed that extracting all semantic information elements from a sentence using a single IE rule (i.e., extracting all instances at the same time) is not efficient because the amount of possible patterns increases largely as the number of

semantic information elements increases. Because some independency exists (but not fully independent) among information elements, extracting information elements separately and sequentially is proposed. The decision regarding the sequence of extraction for different semantic information elements is based on manually analyzing the text and identifying: (1) the level of difficulty for extraction: the easiest semantic information element should be extracted first and the level of difficulty is positively correlated to a combination of the amount of features, the amount of patterns, and the complexity of the patterns; and (2) the existing dependencies across the extractions of the different semantic information elements. For example, (1) if the extraction of “quantity value” only needs the POS tag “CD” as the feature for recognizing cardinal numbers (both appearances of digits and words) and the level of difficulty for its extraction is lowest, then it should be extracted first; and (2) if the extraction of “subject restriction” depends on the extraction of “subject,” then “subject” should be extracted before “subject restriction.” For the example in Figure 3.2, the sequence of extraction of semantic information elements was “quantity value” and “quantity unit/quantity reference” > “subject” > “compliance checking attribute” > “comparative relation” > “quantitative relation” and “deontic operator indicator” > “subject restriction” and “quantity restriction.”

To verify the proposed approach for separation and sequencing of semantic information elements (SSSIE), an experiment was conducted to compare the performance results of two cases. In the first case, IE rules that extract all semantic information elements from a sentence

using a single IE rule (i.e., extracting all instances at the same time) were developed and used. In the second case, the proposed method for SSSIE in information extraction was used. For both cases, the IE rules were developed based on Chapter 12 and 23 of IBC 2006 and were tested using Chapter 19 of IBC 2009. Eighty-seven (87) and 50 patterns were needed for the first and second cases, respectively, indicating that using the proposed SSSIE method reduces the number of needed patterns in IE rules. Table 3.3 shows the comparative results in terms of precision, recall, and F1-measure. The results show significantly stronger performance using SSSIE (the second case). The weaker performance in the first case may be partially attributed to (1) the fact that enumerating all possible patterns based on a limited development text is difficult (if not impossible); and (2) an error in recognizing a single semantic information element in a given IE rule affects the extraction result of the entire IE rule (and, thus, all other information elements in that rule).

Table 3.3 Comparative Testing of Information Extraction Using or Not Using Separation and Sequencing of Semantic Information Elements (SSSIE): Experimental Results for Chapter 19 of IBC 2009

Number of Instances	Subject	Compliance Checking Attribute	Comparative Relation	Quantity Value	Quantity Unit/ Reference	Total
In gold standard	85	45	85	83	85	383
Extracted with SSSIE	85	46	79.5	83	83	376.5
Extracted without SSSIE	55	30	59.5	64	63.5	272
Correctly extracted with SSSIE	80	43	79.5	81	81	364.5
Correctly extracted without SSSIE	48	27	59.5	62	61.5	258
Precision with SSSIE	94.1%	93.5%	100.0%	97.6%	97.6%	96.8%
Precision without SSSIE	87.3%	90.0%	100.0%	96.9%	96.9%	94.9%
Recall with SSSIE	94.1%	95.6%	93.5%	97.6%	95.3%	95.2%
Recall without SSSIE	56.5%	60.0%	70.0%	74.7%	72.4%	67.4%
F1-measure with SSSIE	94.1%	94.5%	96.7%	97.6%	96.4%	96.0%
F1-measure without SSSIE	68.6%	72.0%	82.4%	84.4%	82.8%	78.8%



### 3.2.5 Phase V – Development of Information Extraction Rules

In this phase, a set of rules are developed to automatically execute the information extraction process. The proposed method includes the development and use of two types of rules: rules for extracting single semantic information elements (IE rules) and rules for resolving conflicts in extraction (CR rules). The IE rules recognize target information for extraction, while the CR rules define the strategy for handling conflicts in extraction.

#### 3.2.5.1 Development of Rules for Extracting Single Semantic Information Elements (IE Rules)

The extraction rules (IE rules) utilize pattern matching methods. The left-hand side of the rule defines the pattern to be matched and the right-hand side defines the part of the matched pattern that should be extracted. Both syntactic (POS tags, PSG-based phrasal tags, and gazetteer terms) and semantic (ontology concepts and relations) text features are used in the IE rules patterns. If a concept in the ontology is used in an IE rule, all of its subconcepts are included in the matching as well. For example, in the following IE rule, “building element” is a concept in the ontology: “If ‘building element’ is matched, extract the matched text as an instance for ‘subject.’” When applied to the example in Figure 3.2, this IE rule extracts “habitable rooms” as an instance of “subject” because “habitable room” matches “Habitable\_Room” (a subconcept of “building element” in the ontology). Figure 3.3 shows a sample IE rule (in English) and its corresponding Java coding [using Java Annotation Patterns Engine (JAPE) rules in GATE].



```
quantitative_relation_and_deontic_operator_indicator_extraction - Notepad
File Edit Format View Help
Phase: quantitative_relation_and_deontic_operator_indicator_extraction
Input: Token Lookup MD VB VBN neg VBZ TO VBP VBD
Options: Control = appleT
Rule: quantitative_relation_extraction
// IE Rule #1 for quantitative relation:
// If "MD + VB" is matched, extract the text matched with "VB" as an instance for "quantitative relation".
({MD}) ({VB}):QRel
):QuantitativeRelation
-->
:QuantitativeRelation
{
gate.AnnotationSet matchedQRel=(gate.AnnotationSet) bindings.get("QRel");
Annotation TheQuantitativeRelation=matchedQRel.iterator().next();
gate.AnnotationSet matchedAnns= (gate.AnnotationSet)
bindings.get("QuantitativeRelation");
gate.FeatureMap newFeatures= Factory.newFeatureMap();
newFeatures.put("QuantitativeRelation",TheQuantitativeRelation);
newFeatures.put("rule","QuantitativeRelation");
annotations.add(matchedAnns.firstNode(),matchedAnns.lastNode(),"QuantitativeRelation", newFeatures);
}
```

Figure 3.3 Sample Information Extraction Rule (in English and Java Coding)

To develop these IE rules, the following three tasks are proposed: pattern construction, feature selection, and semantic mapping. For pattern construction, the patterns take the format of a sequential combination of features (e.g., the pattern “NP VP” usually matches a sentence). The construction of such patterns is an iterative, empirical process (using initial manual text analysis, initial pattern construction, testing and results analysis, and testing-based improvement of constructed patterns). Feature selection aims to select all features present in the constructed patterns. In semantic mapping, the extracted information element instances are mapped to their semantic counterparts. For example, as shown in Figure 3.2, the pattern “MD VB” (i.e., POS tags for “modal verb” “verb”) was constructed for the extraction of “quantitative relation,” POS tags were selected as features, “shall have” matched this pattern, “have” was semantically mapped to “has,” and “has” was accordingly extracted as a “quantitative relation” instance.

### 3.2.5.2 Development of Rules for Resolving Conflicts in Extraction (CR rules)

In the proposed method, the rules for resolving conflicts in extraction [conflict resolution (CR) rules] primarily address the following four types of conflict cases: (1) the number of information element instances of a semantic information element in a single sentence is more than the required, (2) the number of information element instances of a semantic information element in a single sentence is less than the required, (3) there is overlap of extraction results for different semantic information elements, and (4) no conflicts, the number of information element instances of a semantic information element in a single sentence is equal to the required. Each type of conflict case may be handled using one of a set of actions. For conflict case 1, one of the following two actions may be used: (1) keep all information element instances; or (2) set priority rules and select the information element instances with higher priority [e.g., set a higher priority for “not less than” comparing with “above” when encountering multiple comparative relation instances. For example, in the sentence part “nonabsorbent surface to a height not less than 70 inches above the drain inlet” (Provision 1210.3 of IBC 2006), the comparative relation instance extracted is only “not less than,” although both “not less than” and “above” are recognized as candidate comparative relation instances]. For conflict case 2, one of the following three actions may be used: (1) set a default information element instance based on domain knowledge (e.g., the default comparative relation instance may be set to “greater\_than\_or\_equal” when no information element instance is extracted. For example, in the sentence “The outside horizontal clear

space measured perpendicular to the opening shall be one and one half times the depth of the opening” (Provision 1203.4.1.2 of IBC 2006), the default “greater\_than\_or\_equal” is used as a comparative relation instance); (2) use the same instance from the nearest sentence/clause (left or right) if those sentences/clauses describe the same content (e.g., in the sentence “The openable area between the sunroom addition or patio cover and the interior room shall have an area of not less than 8 percent of the floor area of the interior room or space, but not less than 20 square feet” (Provision 1203.4.1.1 of IBC 2006), the subject of the first quantitative relation should also be used for the second quantitative relation); or (3) drop this sentence. For conflict case 3, one of the following three actions may be used: (1) delete all overlapping information element instances and keep only the required number, (2) keep all information element instances, or (3) delete some overlapping information element instances and keep more than the required number. For conflict case 4, one action is used: organize all extracted information element instances into a tuple to describe the corresponding requirement. For example, as shown in Figure 3.2, the following CR rule (a conflict case 4) was applied: if one instance exists for each semantic information element (except for subject restriction and quantity restriction, for which the number of instances could be zero or more), organize those instances into a tuple for the corresponding quantitative requirement. For each case, defining which one of the actions should be executed is determined based on the type of conflict pattern. For example, if the subject of a quantitative requirement is a “space,” then the comparative relation is usually “greater\_than\_or\_equal” when missing. The conflict patterns

and corresponding actions are encoded as CR rules.

### **3.2.6 Phase VI – Extraction Execution**

This phase aims to extract the target information element instances from the regulatory text using the rules developed in Phase V. For example, as shown in Figure 3.2, “habitable room” and “net floor area” were extracted as instances of “subject” and “compliance checking attribute,” respectively.

### **3.2.7 Phase VII – Evaluation**

Evaluation is conducted by comparing the extracted information with a gold standard. The gold standard includes all instances of the target information in the regulatory text source and is manually (or semiautomatically with the help of NLP tools) compiled by domain experts. Evaluation is conducted using the following measures: precision, recall, and F1-measure. Precision, here, is defined as the percentage of correctly extracted information element instances relative to the total number of information element instances extracted [Equation (3.1)]. Recall, here, is defined as the percentage of correctly extracted information element instances relative to the total number of information element instances existing in the source text [Equation (3.2)]. A trade-off exists between precision and recall; using either indicator alone is not sufficient. F-measure is defined as a weighted combination of precision and recall (Makhoul et al. 1999) [Equation (3.3)]. In the proposed method,  $\alpha$  is set to 0.5 to give equal weights to precision and recall. If the evaluation results are satisfactory (e.g., the

F1-measure is greater than 90% or a specific value defined by the user), the process may be terminated and the rules (i.e., IE and CR rules) may be considered as final. If the evaluation results are not satisfactory, the phases may be iterated for performance improvement. Performance improvements in later iterations may be achieved by addressing extraction errors in earlier iterations.

$$P = (\text{number of correct information element instances extracted}) / (\text{total number of information element instances extracted}) \quad (3.1)$$

$$R = (\text{number of correct information element instances extracted}) / (\text{total number of information element instances existing}) \quad (3.2)$$

$$F = \frac{P \times R}{(1 - \alpha) \times P + \alpha \times R}, \text{ where } 0 \leq \alpha \leq 1 \quad (3.3)$$

### 3.3 Experimental Testing and Evaluation

An experiment was conducted to validate the proposed algorithm. Evaluating the algorithm (in terms of precision and recall) and achieving satisfactory performance implies the validity of the proposed approach and method. Quantitative requirements were extracted from randomly selected chapters of IBC 2006 and 2009. The information extraction performance of the algorithm was evaluated by comparing the extraction results against a semiautomatically (using NLP tools) developed gold standard.

### 3.3.1 Source Text Selection

IBC was selected because it is the most widely-adopted building code in the United States. IBC 2006 (ICC 2006) and IBC 2009 (ICC 2009) were used. Chapters 12 and 23 of IBC 2006 were randomly selected for development and Chapter 19 of IBC 2009 was randomly selected for testing. The following two main types of requirements in IBC were identified: (1) “quantitative requirement,” which defines the relationship between an attribute of a certain building element/part and a specific quantity value (or quantity range). For example, “Occupiable spaces, habitable spaces and corridors shall have a ceiling height of not less than 7 feet 6 inches (2286 mm)” (Provision 1208.2 of IBC 2006) states that the “ceiling height” attribute of these spaces should be greater than or equal to 7’6”; and (2) “Existential requirement,” which requires the existence of a certain building element/part. For example, “The unit (efficiency dwelling unit) shall be provided with a separate bathroom containing a water closet, lavatory and bathtub or shower” (Provision 1208.4 of IBC 2006) states that an efficiency dwelling unit should have a bathroom with water closet, lavatory, and bathtub or shower. The decision was made to experiment with the extraction of quantitative requirements because: (1) most of the requirements identified in these chapters are quantitative requirements (e.g., on average, quantitative requirements represent 41% of the requirements in Chapters 12 and 23 of IBC 2006 and Chapter 19 of IBC 2009); and (2) the sentences describing quantitative requirements appear more complex than those describing existential requirements, implying that they are more difficult to extract. In Chapters 12 and

23 of IBC 2006, 304 sentences containing quantitative requirements were recognized, forming the development text.

### **3.3.2 Ontology Development**

An application-oriented and domain-specific ontology for buildings was developed. In developing the ontology, a simplified version of the methodology by El-Gohary and El-Diraby (2010) was used. Also, concepts from existing construction ontologies [e.g., the IC-PRO-Onto (El-Gohary and El-Diraby 2010)] and from the Industry Foundation Classes (IFC) (IAI 2007) were reused as applicable/necessary.

The simplified methodology that was used included the following main steps: (1) Defining the domain, purpose, and intended users: The domain of the ontology is building design. The purpose of the ontology is for supporting automated compliance checking of building designs with building codes (i.e., Chapter 12 and 23 of IBC 2006, and Chapter 19 of IBC 2009). The intended users of the ontology are designers and building authorities; (2) Identifying the main concepts in the domain of interest: The main concepts related to building design were identified based on a review of: domain literature [e.g., General Service Administration (GSA) PBS-P100 Facilities Standards for the Public Building Services (2015)], building codes (e.g., international building codes), existing construction ontologies [e.g., the IC-PRO-Onto (El-Gohary and El-Diraby 2010)], and Industry Foundation Classes (IFC) (IAI 2007). Examples of the identified concepts are building element, quantity, material, and space.



Following the principle of minimal ontological commitment (Gruber 1995), the intent was not to cover all concepts in the domain, but to only cover the essential concepts that would enable ACC; (3) Organizing the concepts into a concept hierarchy (taxonomy): The identified concepts were organized into a concept hierarchy, in an iterative manner. The whole hierarchy was checked for consistency after addition of each new concept, and adjusted as needed. For example, wood structural panel sheathing was added as a subconcept of sheathing, but when structural sheathing was added, wood structural panel sheathing was moved down into a subconcept of structural sheathing; (4) Ontology coding: The ontology was coded in web ontology language (OWL) (i.e., .owl format) using the GATE Ontology Editor. OWL was selected because it is the most widely-used semantic Web language; (5) Ontology implementation: the ontology was implemented/applied in information extraction; and (6) Ontology evaluation: An application-oriented ontology evaluation method was used (as discussed in the following paragraphs).

As a result, the developed ontology included 360 concepts arranged in a concept hierarchy. For example, “foundation wall” is a subconcept of the concept “wall,” and “exterior foundation wall” is a subconcept of the concept “foundation wall.” At this phase, the developed ontology includes a concept hierarchy only and is, thus, mainly a taxonomy (not a full ontology). A snapshot of the ontology is included in Figure 3.4.



Figure 3.4 A Snapshot of Part of the Developed Ontology

Because the ontology was developed as an application-oriented ontology, the evaluation of the ontology was conducted in an application-oriented way. Application-oriented ontology evaluation uses the ontology in an application and then evaluates the results of the application (Brank et al. 2005; Salama and El-Gohary 2013b). “This is elegant in the sense that the output of the application might be something for which a relatively straightforward and nonproblematic evaluation approach already exists” (Brank et al. 2005). In this case, the ontology was applied in information extraction, and the ontology was evaluated based on the evaluation results of information extraction (i.e., based on concrete measures of precision and recall).

### **3.3.3 Information Representation**

For building codes, a nine-tuple format was used for intermediate information representation: <Subject, Subject Restriction, Compliance Checking Attribute, Deontic Operator Indicator, Quantitative Relation, Comparative Relation, Quantity Value, Quantity Unit/Reference, Quantity Restriction>.” Following the semantic model of ACC as presented in (Salama and El-Gohary 2013b), the semantic information elements are defined as follows [for further elaboration on the semantic model, including these concepts, the reader is referred to Salama and El-Gohary (2013b)]. A “subject” is an ontology concept; it is a “thing” (e.g., building object, space) that is subject to a particular regulation or norm. A “compliance checking attribute” is an ontology concept; it is a specific characteristic of a “subject” by which its compliance is assessed. A “deontic operator indicator” is an indicator; it matches to (or

indicates) the type of deontic modal operator (i.e., obligation represented by *O*, permission represented by *P*, and prohibition represented by *F*) applicable to the current requirement. A “quantitative relation” defines the type of relation for the quantity. For example, in the sentence “The court shall be increased 1 foot in width and 2 feet in length for each additional story” (Provision 1206.3 of IBC 2006), the quantitative relation is “increase,” which semantically describes that the relation between “width of the court” and “1 foot” is “increased for each additional story.” A “comparative relation” is a relation, such as `greater_than_or_equal`, `less_than_or_equal`, `greater_than`, `less_than`, or `equal`, that is commonly used to compare quantitative values (i.e., comparing an existing value with a required minimum or maximum value). A “quantity value” is a value or a range of values that defines the quantified requirement. A “quantity unit” is the unit of measure for the “quantity value.” A “quantity reference” is a reference to another quantity (which presumably includes a value and a unit). For example, in the sentence “The bearing area of headed anchors shall be not less than one and one-half times the shank area,” “shank\_area” is the “quantity reference.” A “quantity value” + “quantity unit” pair or “quantity value” + “quantity reference” pair forms a “quantity.” A “restriction” places a constraint on the definition of a “subject,” “compliance checking attribute,” “comparative relation,” pair of “quantity value” and “quantity unit,” pair of “quantity value” and “quantity reference,” or the full requirement. A “subject restriction” (and, similarly, “quantity restriction”) places a constraint on the definition of a “subject” (or “quantity”), such as by defining the properties of the “subject”

(or “quantity”). An “exception” defines a condition where the requirement does not apply.

Each extracted requirement (1) has one and only one instance of each of the following semantic information elements: subject, comparative relation, quantity value, and quantity unit/reference; (2) has at most one instance of each of the following semantic information elements: compliance checking attribute, deontic operator indicator, and quantitative relation; and (3) has zero, one, or more instances of each of the following semantic information elements: subject restriction and quantity restriction. Table 3.4 shows examples of the nine-tuple representation.

Table 3.4 Examples of Semantic Information Elements and Information Element Instances

Semantic information element	Extracts of example sentence 1	Extracts of example sentence 2	Extracts of example sentence 3
Requirement	A minimum of 1 inch of airspace shall be provided between the insulation and the roof sheathing.	The minimum net area of ventilation openings shall not be less than 1 square foot for each 150 square feet of crawl space area.	Courts shall not be less than 3 feet in width.
Subject	airspace	ventilation_opening	court
Subject restriction	relation (between, insulation, roof_sheathing)	N/A	N/A
Compliance checking attribute	N/A	net_area	width
Deontic operator indicator	obligation	obligation	obligation
Quantitative relation	provide	N/A	N/A
Comparative relation	greater_than_or_equal	greater_than_or_equal	greater_than_or_equal
Quantity value	1	1	3
Quantity unit/reference	inch	square_foot	feet
Quantity restriction	NA	relation (for_each, 150, square_feet, crawl_space_area)	NA

### 3.3.4 Development of Gold Standard

The gold standard was developed semiautomatically. First, all sentences that include a number (the appearance of both digit form and word form of a number to ensure 100% recall of sentences describing quantitative requirements) were extracted automatically. Subsequently, the author and four other researchers (i.e., the annotators) manually deleted false positive sentences and identified all semantic information element instances for each sentence. The annotation was conducted in four steps: (1) an excel sheet for recording the extracted information element instances was prepared and sent to all annotators; (2) a short 15-minute presentation was given to the annotators to outline the objective of the annotation, explain each semantic information element and demonstrate the extraction of example semantic information element instances (for each semantic information element) from Chapter 12 of IBC 2006; (3) a short 15-minute warm-up and question and answer session was conducted where example sentences from Chapter 12 of IBC 2006 were used to train the annotators in this annotation task and clear up any doubts or confusion. Whenever an annotator asked a question, the answer was broadcasted to all annotators together with the question; and (4) the annotators conducted the extraction task independently in the same session. The inter-annotator agreement between each two annotator was evaluated. Table 3.5 shows the inter-annotator agreement results. A gold standard was then developed based on the agreement between annotators and discrepancy resolution. Two main methods were used for discrepancy resolution: (1) if the majority (i.e., at least three) of the annotators achieved

agreement, then the agreed on annotation was used; (2) if the majority (i.e., at least three) of the annotators did not achieve agreement, then a discussion was conducted until they achieved agreement and the agreed annotation was used. Because of the unambiguous nature of quantitative requirements, along with the well-defined information representation that is used in the proposed method, there was a majority agreement in formulating the gold standard. The annotation guidelines are shown in Appendix A.

Table 3.5 Inter-Annotator Agreement on Chapter 19 of IBC 2009 for Information Extraction

Annotator	A	B	C	D	E	Average annotators
A	-	91%	94%	92%	90%	92%
B	91%	-	91%	89%	85%	89%
C	94%	91%	-	94%	88%	92%
D	92%	89%	94%	-	87%	91%
E	90%	85%	88%	87%	-	88%
Average annotators	92%	89%	92%	91%	88%	90%

### 3.3.5 Tool Selection

Many off-the-shelf tools are available today for supporting various NLP tasks including information extraction, such as the Stanford Parser by the Stanford NLP Group and General Architecture for Text Engineering (GATE) by the University of Sheffield (2013). GATE was selected to implement the information extraction algorithm because (1) GATE has been widely and successfully used in IE, such as in Soysal et al. (2010); and (2) it embeds many other NLP tools in the form of plug-ins, such as the Stanford Parser and OpenNLP tools. The following built-in GATE tools were utilized in the experiments: (1) ANNIE system for

tokenization, sentence splitting, POS tagging, and gazetteer compiling; (2) the built-in morphological analyzer for morphological analysis; (3) the built-in ontology editor for ontology building and editing; and (4) JAPE transducer for writing the IE and CR rules.

### **3.3.6 Applying the Information Extraction Method**

The IE and CR rules were developed based on Chapters 12 and 23 of IBC 2006 and were subsequently tested on Chapter 19 of IBC 2009. The ANNIE Hepple POS Tagger was used to generate POS tag features (Table 3.6 provides a sample). A total of 53 POS tag symbols exist in the set of Hepple POS Tags used. The Penn Treebank phrasal tag labels were used for phrase structure analysis. The following three gazetteer lists were compiled: comparative relation list, unit list, and negation list. In addition, the GATE built-in gazetteer lists of numbers and ordinal were used. Table 3.7 shows the number of patterns, features, and CR rules for Chapters 12 and 23 of IBC 2006. The IE and CR rules (developed based on Chapters 12 and 23 of IBC 2006) are intended to support automated extraction of quantitative requirements from any building code. The rules were applied to Chapter 19 of IBC 2009 for testing and evaluation.



Table 3.6 Sample POS Tags and Phrasal Tags

Part of speech tag/phrasal tag	Meaning
ADVP	Adverb phrase
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
IN	Prepositional or subordinating conjunction
JJR	Comparative adjective
MD	Modal verb
NN	Singular or mass noun
NNS	Plural noun
NP	Noun phrase
PP	Prepositional phrase
QP	Quantifier phrase
RB	Adverb
VB	Base form verb
VP	Verb phrase

Table 3.7 Number of Patterns, Features, and CR rules for Chapters 12 and 23 of IBC 2006

Number	Subject	Subject restriction	Compliance checking attribute	Deontic operator indicator	Quantitative relation	Comparative relation	Quantity value	Quantity unit/reference	Quantity restriction
Extraction patterns	NA	29	NA	10	9	2	24	24	48
Features selected	10(304)*	47	1(99)*	8	7	5	28	31	60
CR rules	2	2	5	0	0	4	8	8	9

\*Number in parenthesis represents subconcepts

Additionally, the IE and CR rules are potentially reusable in extracting quantitative requirements from other types of documents/text. They may be reused as is or adapted/extended based on additional development text. To test the potential reusability of the IE and CR rules developed, they were applied (as is, without any modification) to a different type of text. The following document was randomly selected from the Web, with the

only criterion being that the document contains quantitative requirements: “Procedures (Section 700.4) in traffic cabinet ground rod specifications.” The rules were used to extract quantitative requirements from the randomly selected text, and performance was evaluated against a manually-developed gold standard. Table 3.8 shows the results in terms of precision, recall, and F1-measure. As per Table 3.8, the overall F1-measure is greater than 90%. Considering the selection of this testing text is completely random, the high performance achieved indicates that the developed IE and CR rules well captured the regularity in quantitative requirement-related expressions and thus have a good potential of reusability.

Table 3.8 Testing Reusability of IE Rules and CR Rules

Number of instances	Subject	Subject restriction	Compliance checking attribute	Deontic operator indicator	Quantitative relation	Comparative relation	Quantity value	Quantity unit/reference	Quantity restriction	Total
In gold standard	24	0	18	17	16	13	25	25	6	144
Extracted	24	0	18	17	17	17	24	24	7	148
Correctly extracted	21	0	17	17	11	13	24	24	6	133
Precision	87.5%	NA	94.4%	100.0%	64.7%	76.5%	100.0%	100.0%	85.7%	89.9%
Recall	87.5%	NA	94.4%	100.0%	68.8%	100.0%	96.0%	96.0%	100.0%	92.4%
F1-measure	87.5%	NA	94.4%	100.0%	66.7%	86.7%	98.0%	98.0%	92.3%	91.1%

### 3.3.7 Results and Discussion

Table 3.9 summarizes the information extraction results. For Chapter 19 of IBC 2009, on average, 96.9% (95% confidence interval [95.0%, 98.1%]), 94.4% (95% confidence interval [92.1%, 96.1%]), and 95.6% (95% confidence interval [93.5%, 97.1%]) precision, recall, and F1-measure, respectively, were achieved. When calculating the precision and recall for

“subject restriction” and “quantity restriction” instances, the correctness of extracting one restriction instance is calculated as a ratio of the number of correctly extracted concepts and relations to the total number of concepts and relations in that restriction (because each restriction instance may include multiple concepts and relations). When calculating the precision and recall for “comparative relation” instances, partial extraction correctness for the following comparative relations was considered: “greater than or equal” and “less than or equal.” For example, in the following case, the instance was calculated as “half-correctly extracted,” i.e., 0.5: “above” (greater\_than) was extracted, whereas the gold standard included “at or above” (greater\_than\_or\_equal).

Although only “subject restriction,” “comparative relation,” and “quantity restriction” showed a perfect performance value (100.0% for precision), all precision and recall values were greater than or equal to 90.0% except for the recall of “subject restriction.”

Table 3.9 Experimental Results for Chapter 19 of IBC 2009

Number of instances	Subject	Subject restriction	Compliance checking attribute	Deontic operator indicator	Quantitative relation	Comparative relation	Quantity value	Quantity unit/reference	Quantity restriction	Total
In gold standard	85	18	45	48	58	85	83	85	15	522
Extracted	85	15	46	47	57	79.5	83	83	13.5	509
Correctly extracted	80	15	43	46	54	79.5	81	81	13.5	493
Precision	94.1%	100.0%	93.5%	97.9%	94.7%	100.0%	97.6%	97.6%	100.0%	96.9%
Recall	94.1%	83.3%	95.6%	95.8%	93.1%	93.5%	97.6%	95.3%	90.0%	94.4%
F1-measure	94.1%	90.9%	94.5%	96.8%	93.9%	96.7%	97.6%	96.4%	94.7%	95.6%

An error analysis resulted in five findings. First, the reasons for the relative low recall of

“subject restriction” are as follows: (1) The patterns are more complex. For example, one pattern for “subject restriction” typically involves several phrases, whereas one pattern for other elements such as “subject” could be as simple as corresponding to just one concept in the ontology; and (2) The number of instances for “subject restriction” used in rule development is significantly less (at least 30% less) than that for other types of semantic information elements. These two reasons combined together led to false negatives (i.e., instances that should have been extracted but were not extracted) such as the subject restriction instance “constructed with stud-bearing walls” in the part of sentence “In detached one- and two-family dwellings three stories or less in height and constructed with stud-bearing walls....” (Provision 1908.1.8 of IBC 2009). Second, errors in the extraction of “subject,” which lead to false negatives, are the result of inner errors of the tools used. For example, GATE failed to recognize the term “connection” although it exists in the ontology, which resulted in a false negative of a subject instance “connection.” No existing NLP tool achieves 100% performance, even for relatively simple NLP tasks such as POS tagging, and any error in POS tagging, for example, may further cause an error in information extraction because the IE rules include POS-features in its patterns. Third, errors in extraction of “compliance checking attribute,” which lead to false negatives and positives, are due to inner errors of the tools used and the limitations of CR rules. For example, one CR rule states that if no “compliance checking attribute” was extracted and extra “subject” candidates were extracted, then place the “subject” candidate that is closest to the “quantity value” as the

attribute. This rule lead to an incorrect extraction of “clearance” as the compliance checking attribute instance (i.e., a false positive, meaning it should not be extracted but was extracted) in the sentence “The steel reinforcement shall be in the form of rods, structural shapes or pipe embedded in the concrete core with sufficient clearance to ensure the composite action of the section, but not nearer than 1 inch to the exterior steel shell” (Provision 1915.4 of IBC 2009). Fourth, the errors in the extraction of “deontic operator indicator” and “quantitative relation,” which lead to false negatives, are due to the result of missing patterns in IE rules (which were missed because the patterns are not common) and limitations of CR rules. Fifth, the errors in the extraction of “comparative relation,” “subject restriction,” “quantity restriction,” “quantity value,” and “quantity unit/reference,” which lead to false negatives, are the result of missing patterns in IE rules. Future work is needed to further explore how to improve the proposed IE and CR rules to avoid/reduce these errors, and, consequently, improve the IE results. The problems of missing patterns and limitations of CR rules could be solved through the development/adjustment of IE and CR rules based on more corpuses. However, further exploration is required to find out how many more corpuses could be sufficient to produce enough patterns for IE rules and to avoid the current limitations of the CR rules – and whether the increase in development corpuses would result in significant improvement in precision and recall.

## **4 CHAPTER 4 – AUTOMATED INFORMATION TRANSFORMATION OF REGULATORY INFORMATION**

### **4.1 Comparison to the State of the Art**

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

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

## **4.2 Proposed Information Transformation Method and Algorithm**

The proposed information transformation takes a rule-based, semantic NLP approach. It utilizes pattern-matching-based rules to automatically generate logic rules based on the extracted information instances and their associated patterns of information tags. Both syntactic information tags (i.e., tags tagging syntactic text features, e.g., ‘adjective’ is represented using the POS tag ‘JJ’) and semantic information tags (i.e., tags tagging semantic text features, e.g., ‘compliance checking attribute’ is represented using the semantic tag “a”) are used in defining the patterns. A number of NLP techniques (e.g., POS tagging, term matching) are used to identify the syntactic information tags of each extracted information instance, and a semantic model (an ontology that represents domain knowledge) is used to

identify the semantic information tags. The tagged information instances are transformed into HC-type logic clauses using a set of semantic mapping (SM) rules and conflict resolution (CR) rules. SM rules define how to process the extracted information instances, based on their associated types of information tags and the context of the information tags, so that the extracted information instances could be transformed correctly into logic rules. CR rules resolve potential conflicts that may exist in the processing of different information tags. A bottom-up method is utilized to handle complex sentence components. A “consume and generate” mechanism is proposed to implement the bottom-up method and execute the SM rules. The following subsections present the proposed information transformation method (Figure 4.1) in more detail.

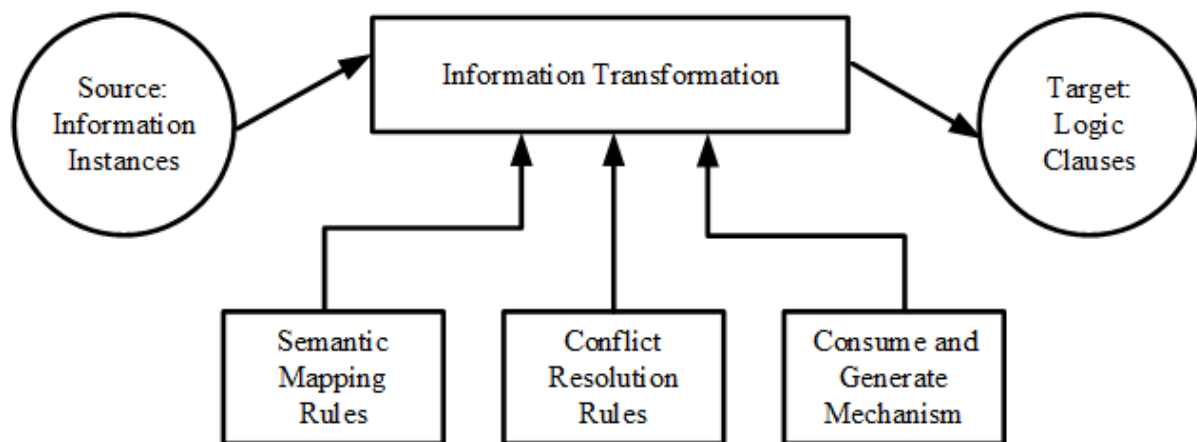


Figure 4.1 The Proposed Information Transformation Method

#### 4.2.1 The Source: Extracted Information Instances

The information source for the information transformation process is the set of input information instances that were obtained from the preceding information extraction process.



Information instances have been labeled with information tags during information extraction, with the following changes/improvements: (1) in addition to semantic information tags, syntactic information tags and combinatorial information tags are also generated for further use in information transformation; and (2) instead of the top-down method for handling complex sentence components (processing larger chunks of texts first, then breaking them down to process smaller chunks of texts), a bottom-up method (processing smaller chunks of texts first, then aggregating them to process larger chunks of texts) is adopted because – in the experiments – it has shown to achieve better performance in handling complex sentence components (Zhang and El-Gohary 2013). As such, in the information transformation process, the following three types of information tags (information tags are shown using single quotes hereafter) are defined and used: (1) semantic information tags, (2) syntactic information tags, and (3) combinatorial information tags.

Semantic information tags are information tags that are related to the meaning and context of the labeled information instances. Instances of semantic information tags are recognized based on the concepts and relations in the domain ontology. For example, in the developed ontology, both “transverse reinforcement” and “vertical reinforcement” are subconcepts of the concept ‘subject’. Therefore, the appearances of “transverse reinforcement” (or “transverse reinforcements”) and “vertical reinforcement” (or “vertical reinforcements”) in Chapter 19 of IBC 2009 are extracted as instances of the semantic information tag ‘subject’. The decision on which concepts and relations are essential to extract and transform is based

on the type of requirement (e.g., quantitative requirements) that is being checked. For example, ‘comparative relation’ is one example of a semantic information tag that is essential in the context of compliance checking of quantitative requirements.

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

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

another example of a combinatorial information tag (combining syntactic and semantic information tags) that describes a ‘subject’ with a certain property.

#### 4.2.2 The Target: Logic Clauses

The target of the information transformation process is the set of output logic rules which are used to represent the requirements in construction regulations. A HC FOL format is used for such representation, in order to facilitate further automated reasoning using logic programs. One single HC represents one requirement. The RHS of the HC indicates the compliance result(s). The LHS of the HC encodes the conditions for the requirement using one or more predicates. Each predicate defines either a concept information instance [e.g., court(c)] or a relation information instance [e.g., has(c,w)]. The logic clause elements in a concept predicate are called concept logic clause elements. The logic clause elements in a relation predicate are called relation logic clause elements. Table 4.1 shows the source and target for a sample sentence.

Table 4.1 A Transformation Example

Requirement sentence	Courts shall not be less than 3 feet in width.					
Source – information tag	Subject	Compliance Checking Attribute	Comparative Relation	Quantity Value	Quantity Unit	Quantity Reference
Source – information instance	court	width	not less than	3	feet	N/A
Target – logic clause	width(width) ^ court(court) ^ has(court,width) ^ greater_than_or_equal(width,quantity(3,Feet)) $\supset$ compliant_width_of_court(court).					

### 4.2.3 Semantic Mapping Rules

The semantic mapping (SM) rules define how to process the extracted information instances according to their semantic meaning. The semantic meaning of each information instance is defined by: (1) the information tag it is associated with. For example, in Table 4.1, ‘subject’ defines the semantic meaning of “court,” i.e., it defines that “court” is the ‘subject’ of compliance checking; and (2) the context of the extracted information instance, reflected by the information tags of its surrounding information instances. For example, in the following sentence, the semantic meaning of “not less than” (instance of ‘comparative relation’) is defined by the information tag of its surrounding information instance “for each”: “The minimum net area of ventilation openings shall not be less than 1 square foot for each 150 square feet of crawl space area” (Provision 1203.3.1 of IBC 2006). “For each,” here, indicates that “not less than” (relation) is not simply a relationship between “net area” (instance of ‘compliance checking attribute’) and “1 square foot” (instance of ‘quantity value’ + ‘quantity unit’), but it is also restricted by “150 square feet of crawl space area” (instance of a ‘quantity value’ + ‘quantity reference’). The interpretation of this requirement is that the quantity requirement on “minimum net area of ventilation openings” will increase 1 foot for each additional “150 square feet of crawl space area.”

The semantic meanings of information instances are utilized in patterns on the LHS of SM rules. For the example in Table 4.1, the corresponding SM rule pattern is ‘subject’ + ‘modal verb’ + ‘negation’ + ‘be’ + ‘comparative relation’ + ‘quantity value’ + ‘quantity unit’ +

‘preposition’ + ‘compliance checking attribute’. An SM rule with this LHS pattern will transform the information instances into the logic clause shown in the last row of Table 4.1. A sample action defined on the RHS of this SM rule is: “Generate predicates for the ‘subject’ information instance, the ‘attribute’ information instance, and a ‘has’ information instance. The two arguments of the ‘has’ information instance are from the ‘subject’ predicate and the ‘attribute’ predicate, respectively.” Accordingly, the following logic clause elements are generated for the following statement, since “court” is recognized as a ‘subject’ information instance and “width” as an ‘attribute’ information instance.

- Sentence: “Courts shall not be less than 3 feet in width” (Provision 1206.3 of IBC 2006)
- Logic Clause Elements:  $\text{court}(\text{court}) \wedge \text{width}(\text{width}) \wedge \text{has}(\text{court}, \text{width})$

The information transformation method is intended to process each term of a sentence in a sequential manner. In general, sequential processing for information transformation normally requires information instances that are matched by patterns (in SM rules) to be strictly located next to each other. Such a rigid processing requirement could cause difficulty in processing sentences with different structures. To avoid that, the proposed SM rules do not follow such a rigid requirement. Instead, the SM rules allow for “look-back searching” (i.e., searching to the left of the matched words) and “look-ahead searching” (i.e., searching to the right of the matched words) to find instances that match certain information tags in a pattern. For example, in the following pattern, the instance of the first ‘subject’ does not have to be

located right next to the instance of ‘preposition’: “ ‘subject’ + ‘preposition’ + ‘subject.’ ” It is only required to be the ‘subject’ instance that is closest to the ‘preposition’ instance from the left. “Look-back searching,” here, searches to the left of the matched word for ‘preposition’ to find the closest ‘subject’ instance when the later part of the pattern “ ‘preposition’ + ‘subject’ ” is matched. This allows for more flexibility in the use of SM rules to handle sentence complexities (e.g., those incurred by cases such as tail recursive nested clauses). For example, an SM rule uses the following pattern *PI* to match the last three information instances in *InSI*, finds the first information instance in *InSI* through “look-back searching,” and generates the logic clause elements *LCI* for the part of sentence *SI*, where ‘s’ stands for ‘subject,’ ‘VBP’ for ‘non-3rd person singular present verb,’ ‘dpvr’ for ‘directional passive verbal relation,’ and ‘VB’ for ‘base form verb:’

- Pattern *PI*: ‘non-3rd person singular present verb’ ‘directional passive verbal relation’ ‘base form verb’
- Information Instances *InSI*: (‘connection’, ‘s’) ... (‘are’, ‘VBP’), (‘designed\_to’, ‘dpvr’), (‘yield’, ‘VB’)
- Sentence *SI*: “Connections that are designed to yield shall be capable of ...”
- Logic Clause Elements *LCI*: connection(connection) ^ yield(yield) ^ designed\_to(connection,yield)

In the proposed method, application-specific SM rules are developed based on a randomly selected sample of text (called “development text,” which is also used for text analysis and

further development of CR rules). For developing a set of SM rules for information transformation, a three-step, iterative method that shall be applied to each sentence is proposed: (1) find all relations in a sentence [e.g., “of” and “not exceed” in the sentence “Spacing of transverse reinforcement shall not exceed 8 inches.” (Provision 1908.1.4 of IBC 2009)]; (2) for each relation, run the existing SM rule set to check if the rule set can generate the corresponding logic clause elements correctly and define the subsequent action based on the following three cases: (a) if the corresponding logic clause elements are correctly generated, then move to check the next relation, (b) if the corresponding logic clause elements are incorrectly generated, then create a new SM rule with a more specific pattern (i.e., a longer pattern with more features) than the applied SM rule and add it to the rule set with a higher priority, and (c) if the corresponding logic clause elements are not generated, then create a new SM rule and add it to the rule set; and (3) after all relations have been checked, run the updated SM rule set on all checked sentences and check if errors have been introduced due to the added SM rules. If errors have been introduced, then identify the source(s) of errors [i.e., the rule(s) that introduced the errors] and adjust those rules as necessary.

#### **4.2.4 Conflict Resolution Rules**

The conflict resolution (CR) rules resolve conflicts between information tags. Two types of CR rules are used: deletion CR rules and conversion CR rules. Deletion CR rules resolve conflicts between information tags by deleting certain information instances. For example,

the following deletion CR rule *CR1* is used to delete redundant information instances *InS2* from the set of extracted information instances *InS3* for the sentence *S2*, where ‘cr’ stands for ‘candidate restriction’ and ‘s’ for ‘subject’:

- Deletion CR Rule *CR1*: “if an information instance has the tag ‘subject’ and it subsumes its following information instance(s), then delete its following information instance(s).”
- Information Instances *InS2*: (‘exterior’, ‘cr’), (‘basement’, ‘cr’), (‘wall’, ‘cr’)
- Information Instances *InS3*: (‘exterior basement wall’, ‘s’), (‘exterior’, ‘cr’), (‘basement’, ‘cr’), (‘wall’, ‘cr’)
- Sentence *S2*: “The thickness of exterior basement walls and foundation walls shall be not less than 71/2 inches.” (Provision 1909.6.1 of IBC 2009)

Conversion CR rules resolve conflicts between information tags by converting information tags of information instances into other types of information tags. For example, the following conversion CR rule *CR2* is used to convert information tags in information instances *InS4* to information tags in information instances *InS5* for the sentence *S3*, where ‘s’ stands for ‘subject,’ ‘I’ for ‘inter clause boundary relation,’ ‘a’ for ‘compliance checking attribute,’ and ‘IN’ for ‘preposition:’

- Conversion CR Rule *CR2*: “if ‘with’ is directly followed by an information instance that has the information tag ‘compliance checking attribute’ and ‘with’ has the



information tag ‘inter clause boundary relation’, then convert the information tag of ‘with’ to ‘preposition’.”

- Information Instances *InS4*: (‘wall segment’, ‘s’), (‘with’, ‘I’), (‘horizontal\_length\_to\_thickness\_ratio’, ‘a’)
- Information Instances *InS5*: (‘wall segment’, ‘s’), (‘with’, ‘IN’), (‘horizontal\_length\_to\_thickness\_ratio’, ‘a’)
- Sentence *S3*: “Wall segments with a horizontal length-to-thickness ratio less than 2.5 shall be designed as columns.” (Provision 1908.1.3 of IBC 2009)

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

CR rule.

#### **4.2.5 Bottom-up Method for Handling Complex Sentence Components**

Due to the variability of natural language expressions and structures, sentences used in regulatory provisions could be very complex. For example, phrases and clauses could be continuously attached/nested to a sentence to constantly enrich it with more relevant information. Complex sentences are difficult to process for information extraction and transformation. Complex sentence components are intermediately-processed segments of text that are: (1) expressed using a variety of natural language structure patterns, and (2) composed of multiple concepts and relations. Complex sentence components are more likely to result in complex sentence structures by embedding in or attaching more concepts and relations to a sentence. Figure 4.2 shows a complex sentence from IBC 2006. Two methods were explored in handling complex sentence components: top-down method and bottom-up method (Figure 4.3). The top-down method starts from the top level (i.e., full sentence) and proceeds down to identify and process complex sentence components. The bottom-up method starts from the lowest level (i.e., single terms/concepts/relations in a sentence) and proceeds up to identify and process complex sentence components. The bottom-up method is employed in the proposed information transformation approach, because it has shown to achieve better performance than the top-down method (Zhang and El-Gohary 2013).

```

A Tagged Sample.txt - Notepad
File Edit Format View Help
Original Text:
Interior spaces intended for human occupancy shall be provided with active or passive space heating systems capable of maintaining a minimum indoor temperature of 68 DegreeF at a point 3 feet above the floor on the design heating day.
Tagged Text:
<s><JJ>Interior</JJ> <cr>spaces</cr></s> <dpvr><VBN>intended</VBN> <IN>for</IN></dpvr> <JJ>human</JJ> <cr>occupancy</cr> <MD>shall</MD> <VB>be</VB> <dpvr><VBN>provided</VBN> <I><IN>with</IN></I></dpvr> <JJ>active</JJ> <CC>or</CC> <JJ>passive</JJ> <s><cr>space</cr> <cr>heating</cr> <cr>systems</cr> </s> <JJ>capable</JJ> <OF><IN>of</IN></OF> <VBG>maintaining</VBG> a <a><ge><JJ>minimum</JJ></ge><JJ>indoor</JJ> <cr>temperature</cr></a> <OF><IN>of</IN></OF> <v>68</v> <u>DegreeF</u> <IN>at</IN> a <cr>point</cr> <v>3</v> <u>feet</u> <g><IN>above</IN></g> the <cr><s>floor</s></cr> <IN>on</IN> the <cr>design</cr> <cr>heating</cr> <cr>day</cr>.
The Tag Legend:
a - Semantic Information Element Tag, for candidate compliance checking attribute
c - Semantic Information Element Tag, for comparative relation
g - Semantic Information Element Tag, for the comparative relation "greater than"
ge - Semantic Information Element Tag, for the comparative relation "greater than or equal"
Has - Semantic Information Element Tag, for "possessing", including the term "has", "have", and "having" so far
OF - Semantic Information Element Tag, indicating "part_of" or "belongs_to" relation by the term "of"
cr - Semantic Information Element Tag, for candidate restriction
s - Semantic Information Element Tag, for candidate subject
u - Semantic Information Element Tag, for a quantity unit
v - Semantic Information Element Tag, for a quantity value
CC - Syntactic Information Element Tag, for conjunctive term, based on POS tag "CC", mainly "and" and "or", the system includes "&" as well
CD - Syntactic Information Element Tag, for cardinal number, based on POS tag "CD"
CHR - Syntactic Information Element Tag, for chapter/section/table/figure reference number, in format of "d.d.d" or "d.d" or "d.d.d.d"
HCJJ - Syntactic Information Element Tag, for hyphen connected adjectives
I - Syntactic Information Element Tag, for the inter clause boundary relation
IN - Syntactic Information Element Tag, for preposition, based on POS tag "IN"
JJ - Syntactic Information Element Tag, for adjective, based on POS tag "JJ"
LS - Syntactic Information Element Tag, for list item marker, based on POS tag "LS"
MD - Syntactic Information Element Tag, for modal verb, based on POS tag "MD"
RB - Syntactic Information Element Tag, for adverb, based on POS tag "RB"
TO - Syntactic Information Element Tag, for literal "to", based on POS tag "TO"
VB - Syntactic Information Element Tag, for base form verb, based on POS tag "VB"
VBD - Syntactic Information Element Tag, for past tense verb, based on POS tag "VBD"
VBG - Syntactic Information Element Tag, for gerund or present participle verb, based on POS tag "VBG"
VBN - Syntactic Information Element Tag, for past participle verb, based on POS tag "VBN"
VBZ - Syntactic Information Element Tag, for 3rd person singular present verb, based on POS tag "VBZ"
WDI - Syntactic Information Element Tag, for wh-determiner, based on POS tag "WDI"
WPP - Syntactic Information Element Tag, for possessive wh-pronoun, based on POS tag "WPS"
WRB - Syntactic Information Element Tag, for wh-adverb, based on POS tag "WRB"
dpvr - Combinatorial Information Element Tag, short for directional passive Verbal relation, for the combination of "past participle verb" (POS tag "VBN") and "preposition" (POS tag "IN")

```

Figure 4.2 A Sample Sentence with Information Tags

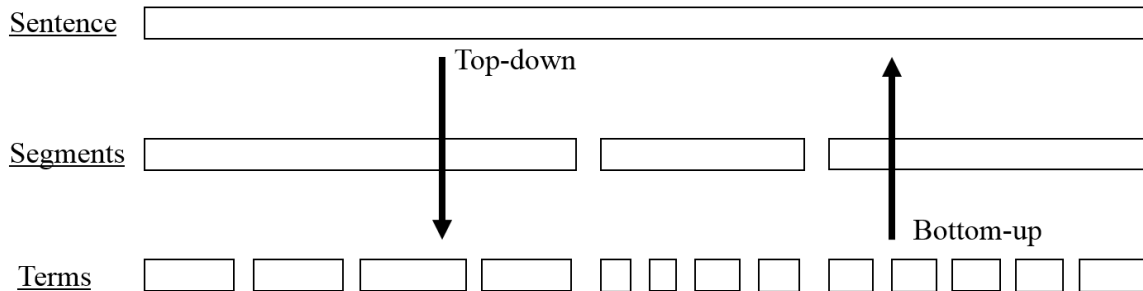


Figure 4.3 Illustration of Top-Down Method and Bottom-Up Method

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

their order in the original sentence.

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

- Pattern *P2*: ‘compliance checking attribute’ ‘of’ ‘subject’
- Information Instances *InS6*: (‘area’, ‘a’), (‘of’, ‘OF’), (‘space’, ‘s’)

- Sentence *S4*: “The net free ventilating area shall not be less than 1/150 of the area of the space ventilated ...” (Provision 1203.2 of IBC 2006)
- Logic Clauses Elements *LC2*:  $\text{space}(\text{space}) \wedge \text{area}(\text{area}) \wedge \text{has}(\text{space}, \text{area})$

The “consume and generate” mechanism allows for backward matching: if information instances extracted from a segment of text match the later part of a pattern, then the information instance(s) extracted from preceding text are checked for matching of the earlier part of the same pattern, and corresponding logic clauses are generated if the check succeeds. For example, the following information tags *InT1* are associated with the five information instances from the part of sentence *S5*. After the first three information instances *InS7* are processed based on matching with the pattern *P3*, two information instances “or” and “space” remain. These two remaining information instances only match the later part (i.e., second and third information tags) of the pattern *P4* for ‘conjunctive subject.’ Normally, this partial matching would not initiate the processing of the information instances. However, under the proposed backward matching mechanism, the preceding information instance “interior room” is checked for the matching of the earlier part of the pattern for “conjunctive subject” (i.e., the first information tag: ‘subject’). Since “interior room” matches ‘subject,’ the SM rule for “conjunctive subject” gets applied and the two remaining information instances are processed to generate the logic clause elements *LC3* [where “v” is the disjunctive operator (i.e., “A v B” means “A or B”)].

- Information Tags *InT1*: ‘compliance checking attribute’, ‘of’, ‘subject’, ‘conjunctive term’, ‘subject’
- Sentence *S5*: “...the floor area of the interior room or space...” (Provision 1203.4.1.1 of IBC 2006)
- Information Instances *InS7*: “floor area,” “of,” “interior room”
- Pattern *P3*: ‘compliance checking attribute’ + ‘of’ + ‘subject’
- Pattern *P4*: ‘subject’ + ‘conjunctive term’ + ‘subject’
- Logic Clause elements *LC3*: interior\_room(Interior\_room) v space(Interior\_room)

#### 4.2.6 Evaluation

Results are evaluated in terms of precision, recall, and F1-measure. Precision, here, is the number of correctly generated logic clause elements divided by the total number of generated logic clause elements. Recall, here, is the number of correctly generated logic clause elements divided by the total number of logic clause elements that should be generated. F1-measure is the harmonic mean of precision and recall, assigning equal weights to precision and recall.

#### 4.3 Experimental Testing and Evaluation

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

### **4.3.1 Source Text Selection**

In alignment with the information extraction work (Chapter 3), IBC 2006 and IBC 2009 were used for testing and evaluation. The SM and CR rules were developed based on Chapters 12 and 23 of IBC 2006, and the proposed information transformation algorithm was tested in processing information instances of quantitative requirements that were extracted from Chapter 19 of IBC 2009 (Chapter 3).

### **4.3.2 Tool Selection**

The proposed information extraction algorithm (Chapter 3) and information transformation algorithm were combined into one computational platform. The representation of Prolog was selected for logic clause representation, in order to facilitate future FOL-based compliance reasoning. Prolog is an approximate realization of the logic programming computational model on a sequential machine (Sterling and Shapiro 1986). It is the most popular logic programming language with a reasoner. The syntax of B-Prolog was used. B-Prolog is a Prolog system with extensions for programming concurrency, constraints, and interactive graphics. It has bi-directional interface with C and Java (Zhou 2012). For information transformation, the SM rules and CR rules were implemented using Python programming language (v2.7.3). The “re” module (i.e., regular expression module) in Python was used for pattern matching, so that each extracted information instance could be used for subsequent processing steps based on their information tags (example tags are shown in Figure 4.2). The

ontology developed in Chapter 3 was used to facilitate semantic information transformation.

### 4.3.3 Information Representation

Each requirement rule in IBC 2006 and IBC 2009 is represented as one single HC-type FOL rule, implemented as a B-Prolog rule. A B-Prolog rule has the form: “H :- B1, B2, ..., Bn. (n>0).” H, B1, ..., Bn are atomic formulas. H is called the head, and the RHS of ‘:-’ is called the body of the rule. A fact is a special kind of rule whose body is always true (Zhou 2012). For the detailed syntax of B-Prolog the reader is referred to Section 7.3.1. Instances of concepts are represented using unary predicates. For example, the information instance “floor” is represented by the predicate “floor(F),” with “floor” being the predicate name and the variable “F” (all variables in B-Prolog start with capitalized letter) being the argument for the predicate. Instances of relations are represented using binary or n-ary predicates. For example, “provided with” is a relation which is represented as the predicate “provided\_with(A,B),” while the variables “A” and “B” could be defined in the predicates interior\_space(A) and space\_heating\_system(B). An example rule is shown in Figure 4.4.



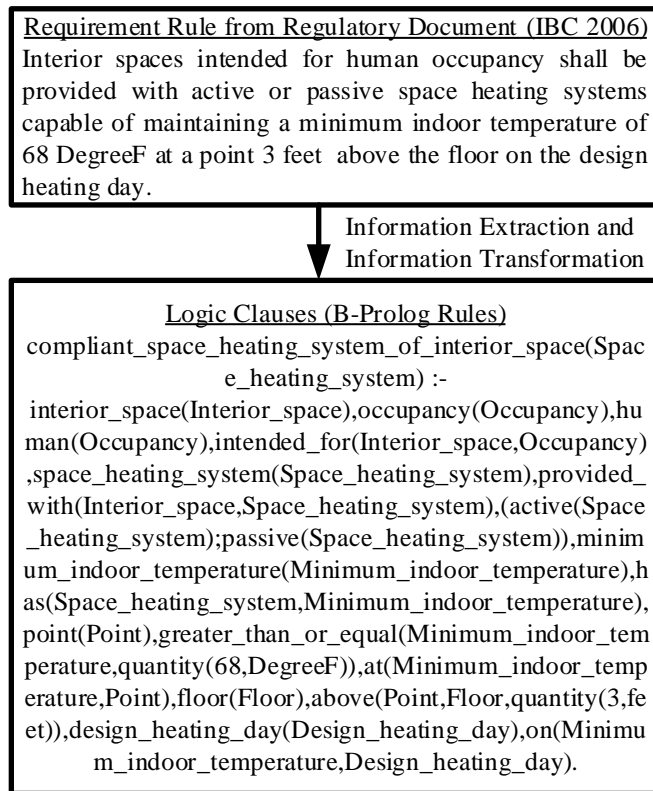


Figure 4.4 An Example Illustrating the Transformed B-Prolog Rule Representation

#### 4.3.4 Information Tags

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

Two main types of semantic information tags were defined (as per Figure 4.5): essential information tags and secondary information tags. Essential information tags are tags for information that must be defined for this specific type of requirement. Six main types of essential information tags were defined for quantitative requirements: subject, compliance

checking attribute, comparative relation, quantity value, quantity unit, and quantity reference (defined in Chapter 3).

Secondary information tags are tags for information that are not necessary for this specific type of requirement, but may exist in defining the requirement. Two main types of secondary information tags were defined for quantitative requirements: restriction and exception. A ‘restriction’ is a concept that places a constraint on the ‘subject,’ ‘compliance checking attribute,’ ‘comparative relation,’ pair of ‘quantity value’ and ‘quantity unit,’ pair of ‘quantity value’ and ‘quantity reference,’ or the full requirement. A ‘subject restriction’ is a concept that places a constraint on the ‘subject.’ Two subtypes of ‘subject restriction’ were further defined: ‘possessive subject restriction’ and ‘nonpossessive subject restriction.’ A ‘possessive subject restriction’ places a possessive constraint on the ‘subject,’ thereby restricting the ‘subject’ to one that possesses certain building parts or properties. For example, in the following requirement sentence, “having windows opening on opposite sides” is a ‘possessive subject restriction’ on “court”: “Courts having windows opening on opposite sides shall not be less than 6 feet in width” (Provision 1206.3 of IBC 2006). A ‘nonpossessive subject restriction’ places a nonpossessive constraint on the ‘subject,’ thereby restricting the ‘subject’ to one that does not possess certain building parts or properties. A ‘compliance checking attribute restriction’ places a constraint on the ‘compliance checking attribute,’ thereby restricting the ‘compliance checking attribute’ to a more specific type. For example, in the following requirement sentence, “to the outdoors” is a ‘compliance checking attribute

restriction' on "minimum openable area": "The minimum openable area to the outdoors shall be 4 percent of the floor area being ventilated" (Provision 1203.4.1 of IBC 2006). A 'comparative relation restriction' places a constraint on the 'comparative relation,' thereby restricting the 'comparative relation' using new conditions. For example, in the following requirement sentence, "for each 150 square feet of crawl space area" is a 'comparative relation restriction' on "not less than": "The minimum net area of ventilation openings shall not be less than 1 square foot for each 150 square feet of crawl space area" (Provision 1203.3.1 of IBC 2006). A 'quantity restriction' places a constraint on the 'quantity value' + 'quantity unit'/'quantity reference' pair, thereby specifying the properties (e.g., range) of the pair. A 'full requirement restriction' places a constraint on the whole quantitative requirement, thereby restricting the quantitative requirement with new preconditions. An 'exception' defines a condition where the described requirement does not apply.

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

One combinatorial information tag was defined for use in this implementation and was called 'directional passive verbal relation', which is the combination of 'past participle verb' (POS tag 'VBN') and 'preposition' (POS tag 'IN'). Combinatorial information tags are expressive

and flexible. Thus, more combinatorial information tags may be defined and used if more complex information tags are needed to capture complex meanings or patterns.

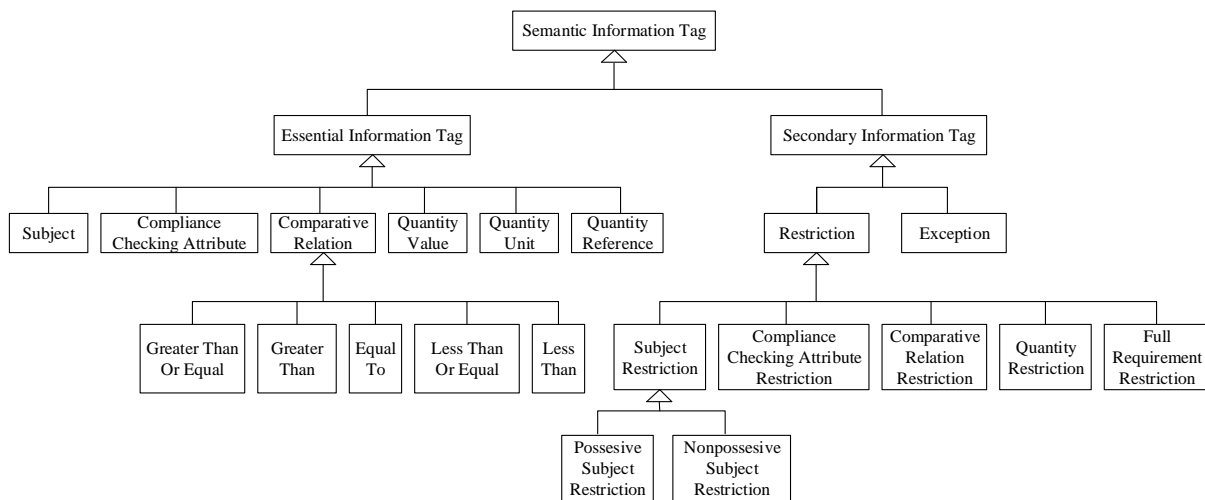


Figure 4.5 Semantic Information Tags

### 4.3.5 Development of Gold Standard

The gold standard for Chapter 19 of IBC 2009 was developed semiautomatically. In the information extraction work (Chapter 3), all sentences that include a number (both appearances of digits and words forms of a number) were automatically extracted to ensure a 100% recall of sentences describing quantitative requirements. Then, the annotators (i.e., the author and four other researchers) manually deleted false positive sentences. After that, the annotators manually coded the logic clauses based on the extracted information instances from each sentence. The annotation was conducted in four steps: (1) an excel sheet for recording the logic clauses was prepared and sent to all annotators; (2) a short 15-minute

presentation was given to the annotators to outline the objective of the annotation, explain concept logic clause element and relation logic clause element, and demonstrate the identification of example concept logic clause elements and relation logic clause elements from Chapter 12 of IBC 2006; (3) a short 15-minute warm-up and question and answer session was conducted where example extracted information instances from Chapter 12 of IBC 2006 were used to train the annotators in this annotation task and clear up any doubts or confusion. Whenever an annotator asked a question, the answer was broadcasted to all annotators together with the question; and (4) the annotators conducted the identification task independently in the same session. The inter-annotator agreement between each two annotator was evaluated. Table 4.2 shows the inter-annotator agreement results. A gold standard was then developed based on the agreement between annotators and discrepancy resolution. Two main methods were used for discrepancy resolution: (1) if the majority (i.e., at least three) of the annotators achieved agreement, then the agreed on annotation was used; (2) if the majority (i.e., at least three) of the annotators did not achieve agreement, then a discussion was conducted until majority annotators achieved agreement and the agreed annotation was used. For Chapter 19, 62 sentences containing quantitative requirements were recognized. Correspondingly, 62 logic clauses were coded. In these 62 logic clauses, 1,901 logic clause elements were identified, including 568 logic clause elements for describing concepts and 1,333 logic clause elements for describing relations between concepts. The annotation guidelines are shown in Appendix B.

Table 4.2 Inter-Annotator Agreement on Chapter 19 of IBC 2009 for Information Transformation

Annotator	A	B	C	D	E	Average annotators
A	-	85%	88%	88%	87%	87%
B	85%	-	86%	92%	84%	87%
C	88%	86%	-	86%	89%	87%
D	88%	92%	86%	-	85%	88%
E	87%	84%	89%	85%	-	86%
Average annotators	87%	87%	87%	88%	86%	87%

#### 4.3.6 Applying the Information Transformation Method

The proposed information transformation method was implemented using Python programming language. The processing steps of an example sentence, the pseudo codes for the main algorithm and the “consume and generate” mechanism are shown in Figure 4.6, Figure 4.7, and Figure 4.8, respectively.

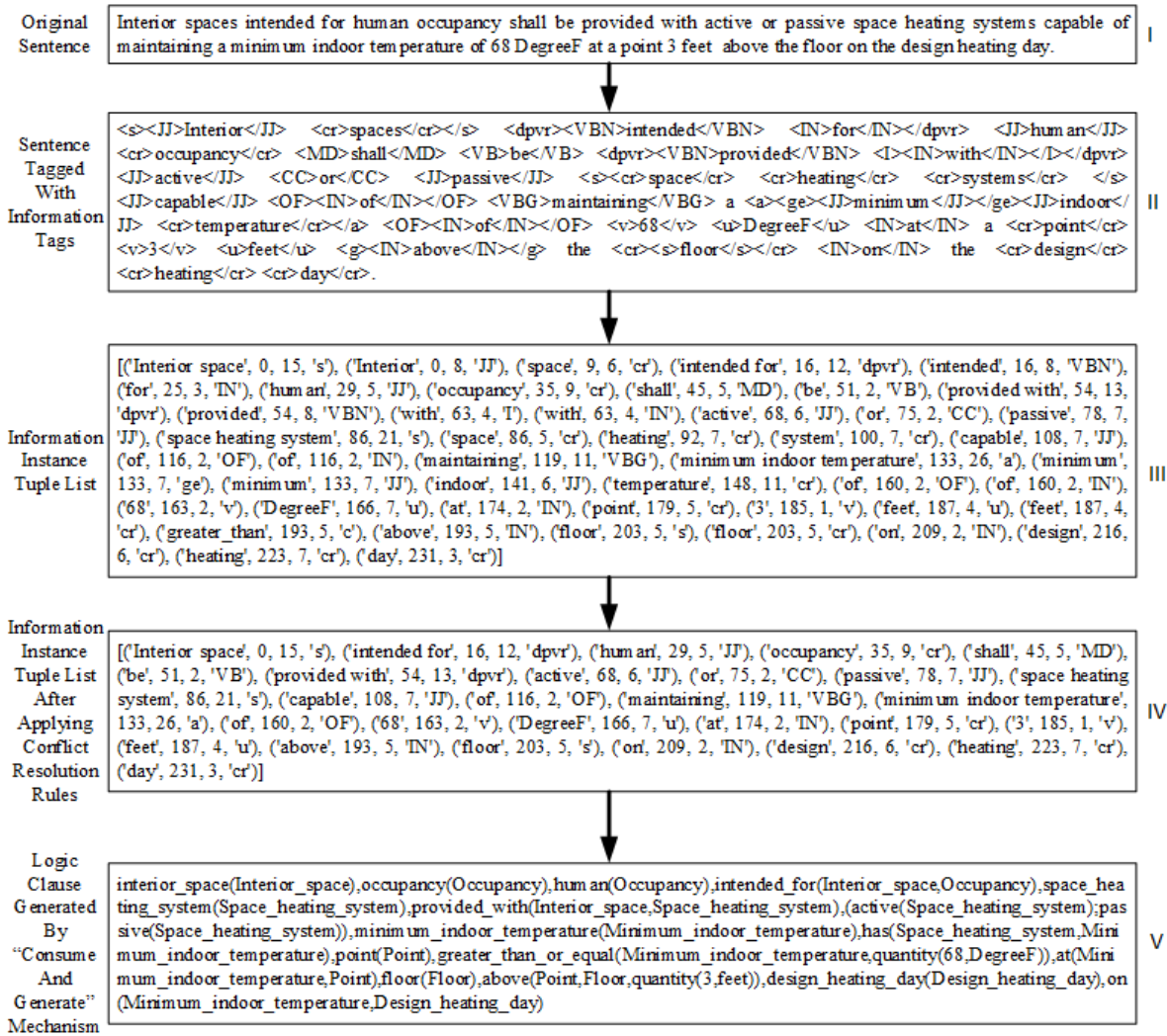


Figure 4.6 An Example Illustrating the Processing of A Sample Sentence

```

open file_tagged_with_information_tags

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

close file_tagged_with_information_tags

```

Figure 4.7 Pseudo Code for Main Information Transformation Algorithm



```

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

```

Figure 4.8 Pseudo Code for “Consume and Generate” Mechanism

As shown in Figure 4.6, the information extraction process tags the original sentence with information tags (from Part I to Part II). The main information transformation algorithm then represents each information instance in the tagged sentence into a four-tuple (from Part II to Part III). The CR rules in the main algorithm then process the information instance tuple list to resolve conflicts between tuples (from Part III to Part IV). The “consume and generate” code then executes the set of SM rules to process each tuple in the list and generate logic clause elements based on matching of SM rule patterns (from Part IV to Part V). For each information instance, the four-tuple is used to store: (1) the information instance itself, (2) the location of the information instance in the corresponding sentence (represented by the starting

point of the information instance in the sentence), (3) the length of the information instance in terms of number of letters, and (4) the information tag of the information instance (e.g., ‘Interior’, 0, 15, and ‘s’ for the first information instance in Part III of Figure 4.6).

In the main algorithm (Figure 4.7), the CR rules are executed through the function “resolve conflicts.” Then, the SM rules are executed using the “consume and generate” code to process the conflict-free information instances for each sentence of the source text file (in the format of a list of four tuples) to generate and display the corresponding logic rule. As shown in Figure 4.8, the “consume and generate” code checks through the patterns for each SM rule (*PATTERN1*, *PATTERN2*, *PATTERN3*...) and generates logic rules as a result of matching to SM rules. In case of no matching, the default negative step length enables backward matching.

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

support mappings from SM patterns to logic clauses.

The logic clauses generated from the SM rules were classified into three main types: single predicate logic clauses, multiple predicate logic clauses, and compound predicate logic clauses. A single predicate logic clause includes only one single predicate [e.g., “space(Space)”]. A multiple predicate logic clause includes more than one predicate [e.g., “space(Space), area(Area), has(Space, Area)”]. A compound predicate logic clause has predicate(s) that embed other predicate(s) as argument(s) [e.g., “greater\_than\_or\_equal(T, quantity(71/2, inches))”].

#### **4.3.7 Results and Discussion**

The proposed information transformation algorithm was tested in transforming information instances of quantitative requirements, which were automatically extracted from Chapter 19 of IBC 2009 (see Chapter 3), into logic rules. The following two experiments were conducted for comparing the performances of two methods of information representation: (1) using essential semantic information tags only, and (2) using both essential and secondary semantic information tags.

In Experiment #1, only the essential semantic information tags were used: ‘subject’, ‘compliance checking attribute’, ‘comparative relation’, ‘quantity value’, ‘quantity unit’, and ‘quantity reference’. A subset of the gold standard (including logic clause elements corresponding to the essential semantic information instances) was used as the gold standard

for Experiment #1. A total of 53 and 11 SM and CR rules, respectively, were developed.

In Experiment #2, both essential and secondary information tags were used. Figure 4.2 shows examples of some of the information tags that were used. A total of 297 and 9 SM and CR rules, respectively, were encoded. The gold standard of Experiment #2 (the full gold standard set) contains 177% more logic clause elements than those in the gold standard of Experiment #1. This shows that for quantitative requirements, the source text contains much secondary information instances.

Table 4.3 shows the patterns of the most applied SM rules (i.e., rules applied at least three times) in the experiments. The patterns of the rest of the applied SM rules are shown in Figure 4.9.

Table 4.3 Patterns of the Most Applied SM Rules in the Experiments

SM rule pattern	Action	Condition case	Logic clause generated	SM rule type
['a' 's' 'cr'] (a) 'OF' (b) ['a' 's' 'cr'] (c)			a(A),c(C),has(C,A)	Simple
'dpvr' (a) ['s' 'cr'] (b)	look-back search for attribute or subject (s); look-back search for negation (n)	n exists	s(S),b(B),not a(S,B)	Complex
		n not exists	s(S),b(B),a(S,B)	
'c' (a) 'v' (b)	look-back search for attribute or subject (s); look-ahead search for unit or reference (u); look-back search for negation (n)	n exists	not a(S, quantity(b,u))	Complex
		n not exists	a(S, quantity(b,u))	
'I' 's'	skip			Multiple action
'c' (a) 'v' (b) 'u' (c) 'IN' (d) 's' (e)	look-back search for attribute or subject (s)		distance(Distance),s(S),e(E), d(S,E,Distance),a(Distance, quantity(b,c))	Multiple action
['a' 's' 'cr'] (a) 'CC' (b) ['a' 's' 'cr'] (c)			(a(A);c(A))	Simple

Table 4.3 (cont.)

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

(1) '': A pair of single quotes encloses information tags

(2) ^: A caret separates optional information tags from exceptions

(3) (a) , (b) , (c) , etc., show the mapping of components (in SM patterns) to logic clause elements (in generated logic clauses), where an upper case represents a variable

(4) Contents in the "logic clause generated" column are case-sensitive

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

Figure 4.9 Patterns of the Rest of the SM Rules Applied in the Experiments

The overall performance results of Experiment #1 and Experiment #2 are summarized in Table 4.4 and Table 4.5, respectively.

Table 4.4 Experimental Results Using Essential Information Tags Only

Measure	Concepts	Relations	Total
Number of logic clause elements in gold standard	334	749	1,083
Total number of logic clause elements generated	328	786	1,114
Number of logic clause elements correctly generated	324	706	1,030
Precision	98.8%	89.8%	92.5%
Recall	97.0%	94.3%	95.1%
F1-measure	97.9%	92.0%	93.8%

Table 4.5 Experimental Results Using Both Essential and Secondary Information Tags

Measure	Concepts	Relations	Total
Number of logic clause elements in gold standard	570	1,349	1,919
Total number of logic clause elements generated	569	1,367	1,936
Number of logic clause elements correctly generated	568	1,333	1,901
Precision	99.8%	97.5%	98.2%
Recall	99.6%	98.8%	99.1%
F1-measure	99.7%	98.2%	98.6%

A comparison between the results of Experiment #1 and those of Experiment #2 is summarized in Table 4.6. The number of information tags in Experiment #2 increased 400% from that used in Experiment #1. The increase in the number of SM rules was of similar magnitude (460%). Through analysis, the causes of this increase in the number of SM rules were found to be: (1) the use of more information tags increases the length of patterns in SM rules, which in turn increases the specificity of each pattern; and (2) the use of more information tags increases the complexity of patterns in SM rules, which in turn increases the possible number of patterns. In contrast to SM rules, the number of CR rules decreased from Experiment #1 to Experiment #2. This results from the use of more information tags, which leads to better distinguishable information instances, and in turn leads to less conflicts between information instances.

Table 4.6 Comparative Summary of Experiment #1 and Experiment #2

Measure	Experiment #1	Experiment #2	Increase
Number of information tags used	8	40	+ 400%
Number of semantic mapping (SM) rules used	53	297	+ 460%
Number of conflict resolution (CR) rules used	11	9	- 18%
Number of logic clause elements built	1,114	1,936	+ 174%
Precision	92.5%	98.2%	6%
Recall	95.1%	99.1%	4%
F1-Measure	93.8%	98.6%	5%

The algorithm achieved 92.5% (95% confidence interval [90.8%, 93.9%]) and 98.2% (95% confidence interval [97.5%, 98.7%]), 95.1% (95% confidence interval [93.7%, 96.2%]) and 99.1% (95% confidence interval [98.5%, 99.4%]), and 93.8% (95% confidence interval [92.2%, 95.0%]) and 98.6% (95% confidence interval [98.0%, 99.0%]) overall precision, recall, and F1-measure for Experiment #1 and Experiment #2, respectively. Both precision and recall improved in Experiment #2, because the use of more information tags could: (1) better distinguish and capture the variations in expressions; and (2) help define SM rules with more specificity in patterns. Based on the comparative analysis, the following conclusion can be drawn: the use of more information tags helps in improving the performance of information transformation.

The precisions of relation logic clause elements are lower than other precision and recall values across Experiment #1 and Experiment #2. Through analysis, four main causes for this relatively lower performance of precision (89.8% and 97.5% for Experiment #1 and Experiment #2, respectively) of relation logic clause elements are recognized: (1) Structural ambiguity caused by conjunctive terms: For example, in the following part of sentence, there



are two possible syntactic uses of “and” – either linking “wall piers” and “such segments” or linking the preceding clause and the following clause: “...shear wall segments provide lateral support to the wall piers and such segments have a total stiffness...” (Provision 1908.1.3 of IBC 2009). The ability of the SM rules to handle structural ambiguity is limited by the development text, which may lead to both false positive and false negative errors; (2) errors from incorrect tagging during IE: For example, “professional” (in “registered design professional”) was incorrectly tagged as an adjective instead of noun and resulted in a false negative instance. This is due to the imperfection of state-of-the-art NLP methods and tools; and (3) Errors caused by certain SM rules: For example, an SM rule selects the immediate left neighbor of a preposition as the first argument of that preposition. In cases where the immediate left neighbor of a preposition is not its real first argument, this SM rule causes errors. For example, in the following part of sentence, “gypsum concrete” was mistakenly identified as the first argument rather than “clear span” which resulted in both a false positive and a false negative: “clear span of the gypsum concrete between supports” (Provision 1914.2 of IBC 2006).

## 5 CHAPTER 5 – SEMIAUTOMATED IFC EXTENSION

### 5.1 Comparison to the State of the Art

Among the existing construction regulatory ACC efforts, building information models (BIMs) were mostly utilized as the representation of design information (Eastman et al. 2009). Due to the lack of a fully developed all-inclusive BIM data/information schema that can sufficiently represent project information for ACC needs in different areas (e.g., fire safety, structural safety, and sustainability), existing ACC efforts typically went into one of two directions for preparing BIMs: either creating their own BIM or extending existing BIMs.

One of the important ACC projects, the Construction and Real Estate NETWORK (CORENET) project of Singapore (Khemlani 2005), developed their own semantic objects in FORNAX library (i.e., a C++ library) to represent building design information. In the U.S., the General Services Administration (GSA) design rule checking efforts defined the BIM modeling requirements in a well-documented building information modeling guide and allowed users to choose their own BIM authoring tool to define building models according to the guide (Eastman et al. 2009). In addition, many of the existing research efforts proposed or implemented the idea of extending BIMs to fulfill their specific information needs. For instance, Nguyen and Kim (2011) and Sinha et al. (2013) extended existing BIMs in Autodesk Revit Architecture by creating new project parameters such as “area of opening in firewall” and “width of opening in firewall;” Kasim et al. (2013) extended existing BIMs through adding new data items into IFC-represented BIMs directly; Nawari (2011) proposed

the development of appropriate information delivery manuals (IDM) and model view definitions (MVDs) for the ACC domain to achieve the required level of detail on IFC-represented BIMs; and Tan et al. (2010) extended IFC in eXtensible markup language (ifcXML) to develop an extended building information modeling (EBIM) in XML.

These existing efforts to extend BIMs for ACC deepened the understanding of BIM modeling requirements for ACC. However, the model extension methods were mostly ad-hoc and subjective (i.e., relying on subjective developments or extensions by individual software developers and/or researchers); and the resulting models were usually still missing essential compliance checking (CC)-related information that are needed to achieve complete automation in CC (Martins and Monteiro 2013; Niemeijer et al. 2009). In addition, such ad-hoc and subjective developments/extensions lack generality and objectivity, which are essential to full automation of CC at a broader scale. As a result, a more generalized and objective method is needed to extend BIMs for facilitating ACC. Despite the potential of using ontology alignment and ontology mapping techniques (using semantic similarities) in developing a generalized IFC extension method, to the best of the author's knowledge there was little empirical exploration of this approach. The work of Delgado et al. (2013) and Pan et al. (2008) are the closest to this approach.

Delgado et al. (2013) evaluated 15 ontology matching techniques in matching geospatial ontologies with BIM-related ontologies (including an ontology for IFC) to discover correspondences of concepts between each pair of ontologies [e.g., between City Geography

Markup Language (CityGML) ontology and IFC ontology]. The 15 techniques were classified into three categories: string-based techniques, WordNet-based techniques, and matching systems techniques. The alignment between CityGML ontology and IFC ontology is conceptually and technically similar to extending the IFC. In their experimental results: (1) string-based techniques showed the best performance [100% precision (i.e., the number of correctly found correspondences divided by the total number of correspondences found), 57.1% recall (i.e., the number of correctly found correspondences divided by the total number of correspondences that should be found), and 23.2% F1-measure (i.e., the harmonic mean of precision and recall)] among the three tested techniques; (2) within the WordNet-based techniques, the synonym distance technique showed 41.6% precision, 14.2% recall, and 21.2% F1-measure; and (3) within the matching systems techniques, the association rule ontology matching approach (AROMA) showed 40% precision, 5.7% recall, and 10% F1-measure. These results show that further research is needed to investigate whether the use of other semantic relations in WordNet (such as hyponymy), in addition to synonymy, would result in higher levels of performance.

Pan et al. (2008) conducted semiautomated mapping of AEC ontologies, including an IFC ontology, using relatedness analysis techniques. In their ontology mapping, three types of features were used to provide expert guidance: (1) corpus-based features: co-occurrence frequencies between two concepts, (2) attribute-based features: attribute value structures of ontologies, and (3) name-based features: stemmed terms of the concept names to use for

direct term-based matching. Further research is needed to explore how different types of semantic relations among concepts could be leveraged in IFC concept mapping.

## 5.2 Proposed IFC Extension Method and Algorithm

The proposed method for semiautomated IFC extension with regulatory concepts includes four primary phases (Figure 5.1): regulatory concept extraction, IFC concept selection, relationship classification, and regulatory concept integration. The proposed method is semiautomated, where concept extraction, concept matching and similarity assessment for IFC concept selection, relationship classification of concepts, and regulatory concept integration into IFC class hierarchy are conducted automatically and the user checks the results of each phase and removes/fixes errors as necessary.

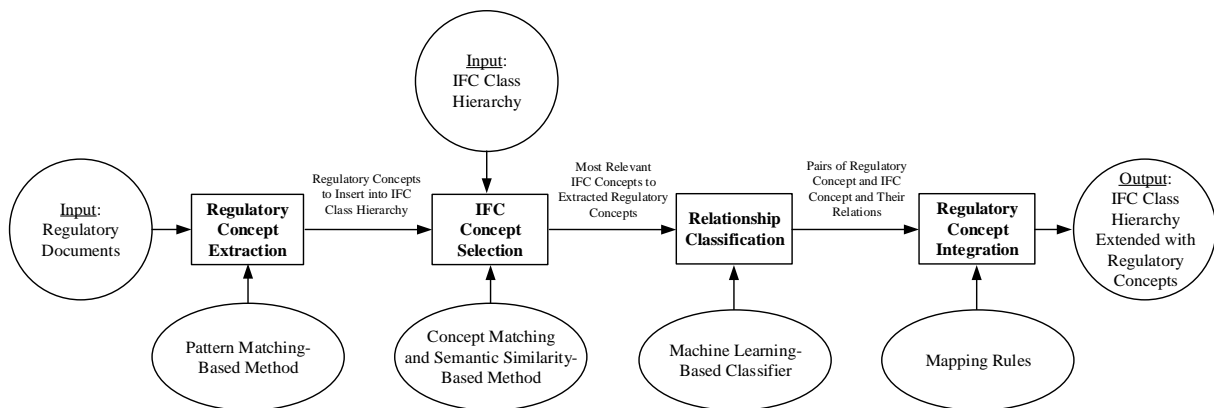


Figure 5.1 The Proposed IFC Extension Method

### 5.2.1 Regulatory Concept Extraction

#### 5.2.1.1 Proposed Concept Extraction Approach

To conduct ACC in a fully automated way, all concepts related to regulatory requirements in

a relevant regulatory document must be incorporated into a BIM schema (e.g., IFC schema). The regulatory concept extraction phase aims to automatically extract all concepts from a selected, relevant regulatory document. The proposed extraction method utilizes pattern-matching-based extraction rules. After all concepts are automatically extracted from a textual regulatory document, they get displayed to the user, where the user can review all concepts and manually remove those concepts that are incorrect or irrelevant to regulatory requirements. The extent of manual effort is minimal, as it only requires a review of the extracted concepts to remove clearly incorrect/irrelevant concepts (e.g., given these two concepts, the “reinforced” is incorrect, while “reinforced gypsum concrete” is the correct concept).

#### 5.2.1.2 Concept Extraction Rules

The concept extraction rules are pattern-matching based. Each concept extraction rule has a LHS and a RHS. The LHS of a rule defines the pattern to be matched and the RHS defines the concept that should be extracted. The patterns are composed of POS features (i.e., POS tags). For example, Figure 5.2 shows an example concept extraction (CE) rule for extracting four-term concepts like “thermally isolated sunroom addition.” Ten selected POS tags from Penn Treebank tag set (Santorini 1990) are also listed in Figure 5.2. Only flattened patterns are utilized in the CE rules to avoid recursive parsing. Flattened patterns are patterns that include only terminal symbols (i.e., symbols that cannot be further broken down), which are analogous to leaf nodes in a tree-like structure. For example, in Table 5.1, the patterns P1, P2,

P3, and P4 are flattened because they only contain POS tags (i.e., “NN,” which is the POS tag for singular noun). Non-flattened patterns, on the other hand, are patterns that include non-terminal symbols (i.e., symbols that could be further broken down). For example, in Table 5.1, the “NN NP” pattern in rule R2 is non-flattened because it contains non-terminal symbols (i.e., “NP,” which is a phrase level tag for noun phrase that could be further broken down). Recursive parsing is avoided, in the proposed method, to enhance computational efficiency and matching flexibility, because: (1) recursive parsing increases time complexities of parsing algorithms. For example in Table 5.1, to match the pattern P4, the number of trials for applying rules R1 and R2 using recursive parsing are minimum 4 and maximum 8, higher than the number of trials for applying rules R1, R3, R4, and R5 using non-recursive parsing which are minimum 1 and maximum 4; and (2) recursive parsing is less flexible. For example, if only P1 and P3 should be matched while P2 and P4 should not, this is easy to achieve through applying R1 and R4 using non-recursive parsing, whereas it is difficult to achieve using recursive parsing.

**Concept Extraction Rule:** “RB” “VBN” “NN” “NN” -> Extract the four matched terms

**Meaning:** If four consecutive terms are adverb, past participle verb, singular/mass noun, and singular/mass noun, then these four terms should be extracted as a concept.

POS Tag	Meaning
NN	Singular or mass noun
NNS	Plural noun
NNP	Singular proper noun
NNPS	Plural proper noun
JJ	Adjective
RB	Adverb
VBN	Past participle verb
VBP	Non-3rd person singular present verb
VBD	Past tense verb
VBG	Gerund or present participle verb

Figure 5.2 An Example CE Rule and Its Meaning

Table 5.1 Sample of Patterns and Concept Extraction Rules

Pattern/Rule Number	Pattern/Rule
P1	NN NN
P2	NN NN NN
P3	NN NN NN NN
P4	NN NN NN NN NN
R1	NP → NN NN
R2	NP → NN NP (non-flattened)
R3	NP → NN NN NN
R4	NP → NN NN NN NN
R5	NP → NN NN NN NN NN

### 5.2.1.3 Development of POS Pattern Set

The development of the set of POS patterns to use in the CE rules is conducted based on “development text,” following the algorithm shown in Figure 5.3. A development text is a



sample of regulatory text (Chapter 12 of IBC 2006 was used in this dissertation) that is used to identify common POS patterns in the text for developing the POS pattern set. The algorithm is executed after (1) the gold standard of regulatory concepts for the development text (i.e., a list of all regulatory concepts in the development text) is created, and (2) the POS tags for all sentences in the development text are generated. The algorithm incrementally processes concepts in the gold standard using two levels of loops: the outer loop accesses each sentence in the gold standard and the inner loop accesses each concept in the sentence being accessed. In the processing of each concept, the POS pattern for the concept is first tentatively collected into the POS pattern set. Then the POS pattern set is used to extract concepts from all sentences. The precision (i.e., the number of correctly extracted concepts divided by the total number of extracted concepts), recall (i.e., the number of correctly extracted concepts divided by the total number of concepts that should be extracted), and F1-measure (harmonic mean of precision and recall) are then calculated for the result. If the recall and F1-measure increase comparing to the previous recall and F1-measure (without the tentatively added POS pattern), then the addition of the POS pattern into the POS pattern set is committed. This process iterates through all concepts in all sentences. The algorithm iteratively improved recall and F1-measure of extraction by incorporating more POS patterns.

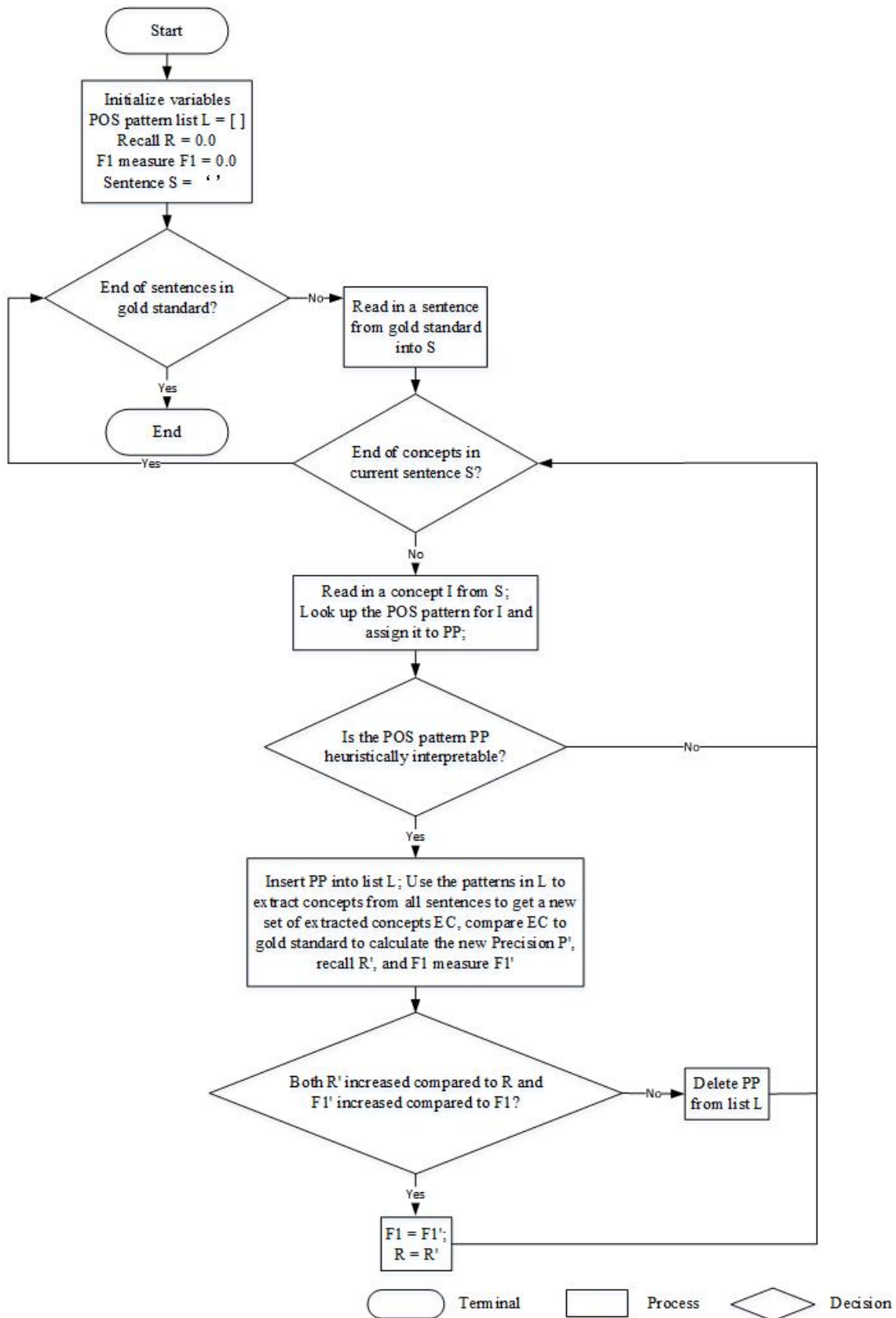


Figure 5.3 Flow Chart of the POS Pattern Set Development Algorithm

#### 5.2.1.4 Exclusion Word Removal

Exclusion words are defined, here, as words (unigram, bigram, or multigram) that match certain POS patterns in the CE rules but should not be extracted as concepts. The POS tags of these exclusion words usually introduce ambiguity because they carry more than one lexical or functional category/meaning, which may introduce false positives (i.e., incorrectly extracted concepts) in concept extraction. For example, “VBG” (POS tag for both “verb gerund” and “present participle”) is useful to extract “verb gerund” concepts like “opening,” but it introduces false positives when incorrectly extracting “present participle” words like “having” as concepts. To avoid introducing false positives during concept extraction, an exclusion word list is used.

### **5.2.2 IFC Concept Selection**

#### 5.2.2.1 Proposed Concept Selection Approach

The IFC concept selection phase aims to (1) automatically find the most related concept(s) in the IFC schema [called F-concept(s) hereafter] to each extracted regulatory concept (called R-concept hereafter) and (2) accordingly, allow the user to select the F-concept(s) for each R-concept. In the targeted IFC extension method, the extension of the IFC schema is an incremental process; each R-concept is added to the IFC schema one by one, incrementally. As a result, an R-concept that gets selected (and thus added) to the IFC schema becomes part of the schema (i.e., becomes an F-concept for the following automated selection step). The automated IFC concept selection method includes four steps/techniques (as shown in Figure

5.4): (1) Step 1: stemming, which reduces words to their stems (i.e., base or root form); (2) Step 2: term-based matching, which aims to find all F-concepts that share term(s) with an R-concept; and (3) Step 3: semantic-based matching, which aims to find all semantically related F-concepts to an R-concept. Semantic-based matching is used, to add a deeper level of searching, if the term-based matching fails to find candidate concepts; and (4) Step 4: semantic similarity (SS) scoring and ranking, which measures the SS between each candidate F-concept (from Step 2 and Step 3) and the R-concept, and accordingly ranks all candidate F-concepts related to that one single R-concept for final F-concept user selection. The same process is repeated for all R-concepts and their related candidate F-concepts.

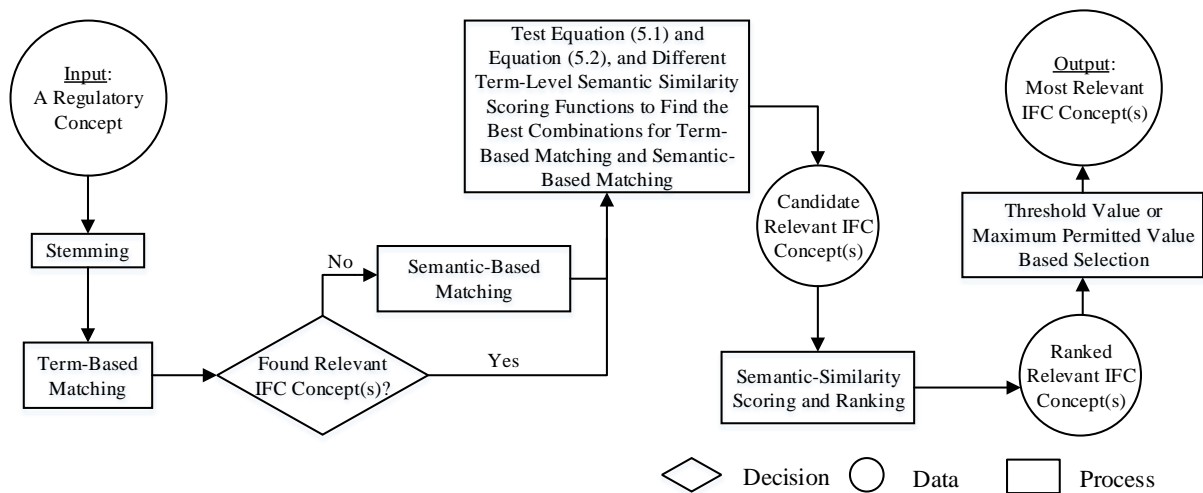


Figure 5.4 Steps for IFC Concept Selection Method

### 5.2.2.2 Stemming

Stemming is utilized in both term-based and semantic-based matching. Concepts are stemmed before matching to avoid incorrect mismatching due to variant word forms (rather than variant meaning). For example, with stemming applied, “foot” could be matched to “feet”

(the stem of “feet” is “foot”).

### 5.2.2.3 Term-Based Matching

For term-based matching, three types of matching are used, based on the following two heuristic rules, H1 and H2: (1) “First Term Term-Based Matching”: the first term in the R-concept is terminologically matched against all F-concepts to find related F-concepts, (2) “Last Term Term-Based Matching”: the last term in the R-concept is terminologically matched against all F-concepts to find related F-concepts, and (3) “First and Last Term Term-Based Matching”: the first and last terms in the R-concept are terminologically matched against all F-concepts to find related F-concepts. Which type of matching to use depends on two main factors: (1) the number of terms in the concept name of the R-concept, whether the concept name is unigram (i.e., concept name with only one term), bigram (i.e., concept name with two terms), or multigram (i.e., concept name with three or more terms); and (2) the types of POS patterns in the concept name of the R-concept, whether the pattern is “N” (i.e., a POS pattern with only one POS tag and the POS tag is a noun), “NN” (i.e., a POS pattern starting with a noun and ending with a noun), or “JN” [i.e., a POS pattern starting with a pronominal modifier (e.g., adjective) and ending with a noun]. The matching strategy is illustrated in Figure 5.5. For example, the R-concept “interior space” is a bigram and its POS pattern (“JJ NN”) matches “JN;” therefore, last term matching is used to find matching concepts that contain the term “space” (i.e., the last term in the R-concept).

- H1: The term that has a nominal POS tag (i.e., noun) is the primary meaning-carrying term in a multi-term concept name.
- H2: The terms that have non-nominal POS tags (e.g., “JJ”) are the secondary meaning-carrying terms in a multi-term concept name, which add to or constrain the meaning as modifiers.

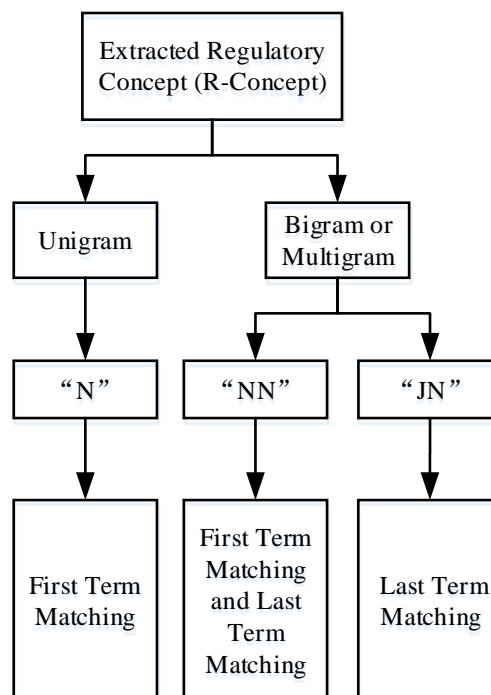


Figure 5.5 Term-Based and Semantic-Based Matching Strategy

#### 5.2.2.4 Semantic-Based Matching

In semantic-based matching, the semantic relations of WordNet (Fellbaum 2005) are utilized to find concept matches beyond term-based matching. Three types of these relations are used: hypernymy, hyponymy, and synonymy. These three types were selected because they are most relevant to the superclass-subclass structure of the IFC class hierarchy. Hypernymy is a

semantic relation where one concept is the hypernym (i.e., superclass) of the other. For example, “room” is a hypernym of “kitchen.” Hyponymy is the opposite of hypernymy where one concept is the hyponym (i.e., subclass) of the other. For example, “kitchen” is a hyponym of “room.” Synonymy is the semantic relation between different concepts who share the same meaning. For example, “gypsum board,” “drywall,” and “plasterboard” all share the same meaning of “a board made of gypsum plaster core bonded to layers of paper or fiberboard.” Three types of matching are used, which semantically match the first term, the last term, or the first term and last term in the R-concept, respectively, against all F-concepts to find related F-concepts: “first term semantic-based matching,” “last term semantic-based matching,” and “first and last term semantic-based matching.” To conduct the semantic matching, the hypernyms, hyponyms, and synonyms of the first/last term are determined, based on WordNet, and then term-matched against all F-concepts to find related F-concepts. Similar to term-based matching, which type of matching to use depends on (1) the number of terms in the concept name of the R-concept and (2) the POS pattern types in the concept name of the R-concept. The matching strategy is illustrated in Figure 5.5.

#### 5.2.2.5 Semantic Similarity Scoring and Ranking

The proposed SS scoring method follows heuristic rules H3, H4, and H5.

- H3: In a multi-term concept name, the contribution of each term’s carried meaning to the meaning of the whole concept decreases from right to left; the first term contributes the least to the meaning of the whole concept.

- H4: The length of a concept name is related to its level in a concept hierarchy. The shorter the length of a concept name is, the more general the concept is; and thus the higher its level in a concept hierarchy. The longer the length of a concept name is, the more specific the concept is; and thus the lower its level in a concept hierarchy. A superconcept is, thus, likely to have a shorter concept name length than its subconcept.
- H5: The difference in length between two concept names (where the length is measured in number of terms) is indicative of the closeness of the two concepts in a concept hierarchy; the smaller the difference, the closer the two concepts are, and vice versa. Sibling concepts are, thus, likely to have a small difference between their concept name lengths.

Based on these heuristic rules, Equation (5.1) and Equation (5.2) are proposed as two alternative functions for SS scoring, where  $SS_{RF1}$  and  $SS_{RF2}$  are the concept-level SS scores between an R-concept and an F-concept,  $SS_{RmFk}$  is the term-level SS score between the  $m^{\text{th}}$  term in the R-concept and the  $k^{\text{th}}$  term in the F-concept,  $m$  is the ordinal number for the term  $Rm$  in R-concept,  $k$  is the ordinal number for the term  $Fk$  in F-concept,  $L_F$  is the length of F-concept measured in number of terms, and  $L_R$  is the length of R-concept measured in number of terms.

$$SS_{RF1} = \frac{1}{L_F} \sum_{k=1}^{L_F} \frac{2k}{L_F(L_F+1)} SS_{RmFk} \quad (5.1)$$



$$SS_{RF2} = \frac{1}{|L_R - L_F| + 1} \sum_{k=1}^{L_F} \frac{2k}{L_F(L_F + 1)} SS_{RmFk} \quad (5.2)$$

Any existing term pair SS measure, such as the Shortest Path Similarity measure or the Leacock-Chodorow Similarity measure, can be used (after testing) to compute  $SS_{RmFk}$ . In Equation (5.1) and Equation (5.2), each term-level SS score (i.e.,  $SS_{RmFk}$ ) is discounted using

the factor  $\frac{2k}{L_F(L_F + 1)}$ . This term-level discount factor is based on heuristic rule H3. The

concept-level SS score between the R-concept and the F-concept (i.e.,  $SS_{RF}$ ) is determined by further discounting the summation of all discounted term-level SS scores (of all term pairs formed between the matching term of R-concept and each term of the F-concept). In Equation (5.1), the concept-level discount factor is  $\frac{1}{L_F}$ , which linearly discounts the

summation using the length of the F-concept. This discount favors concepts at higher levels in a concept hierarchy and follows heuristic rule H4 to identify higher-level concepts based on the lengths of concept names. In Equation (5.2), the concept-level discount factor is

$\frac{1}{|L_R - L_F| + 1}$ , based on the absolute length difference between the concept names of R-concept

and F-concept. This discount favors concepts at similar levels in a concept hierarchy and follows heuristic rule H5.

Accordingly, the proposed SS scoring method is summarized in Figure 5.4. Combinations of different concept-level SS scoring functions [i.e., Equation (5.1) and Equation (5.2)] and term-level SS scoring functions (i.e., existing similarity measures such as Shortest Path Similarity) should be experimentally tested to select the best-performing combination.

Separate testing is conducted for term-based matched F-concepts (i.e., F-concepts found using term-based matching, from Step 2) and semantic-based matched F-concepts (i.e., F-concepts found using semantic-based matching, from Step 3). The experimental testing and results are presented and discussed in the Experimental Testing and Evaluation section (Section 5.3).

For SS ranking, all candidate F-concepts related to one single R-concept are ranked according to their SS scores, in order of decreasing score. A threshold value or a maximum permitted value is further used to filter the most related F-concept(s) among the candidate concepts. The threshold is the minimum SS score below which a candidate F-concept is considered semantically not related (and thus ineligible for selection for this R-concept). The maximum permitted value is a natural number (default is 1) that defines at most how many number of F-concepts could be selected for a single R-concept. Both, threshold value and maximum permitted value, are set by the user. For example, using term-based matching, a number of F-concepts were found to match “exterior wall” through the matching term “wall,” such as “wall” and “curtain wall.” Then, the SS scores were computed between “exterior wall” and each of the matched F-concepts, such as “wall” and “wall,” and “wall” and “curtain wall.” The candidate F-concepts were ranked according to the SS scores and the highest scored candidates were automatically selected, according to the default maximum permitted value. If the maximum permitted value is set to 1 and Equation (5.1) is used, “wall” is selected because of its highest SS score. Following a similar process, but using semantic-based

matching, “window” was selected as the match to “skylight.”

### 5.2.3 Relationship Classification

#### 5.2.3.1 Proposed Classification Approach

The relationship classification phase aims to classify the relationship between each pair of R-concept and F-concept. ML techniques are used to automatically predict the relationship between a concept pair based on the concept features of the pair.

#### 5.2.3.2 Types of Relationships

Four types of relationships are considered (Table 5.2): (1) equivalent concept, indicating that the R-concept and the F-concept are equivalent (e.g., “diameter” and “diameter dimension”); (2) superconcept, indicating that the R-concept is a superconcept of the F-concept (e.g., “lighting” and “surface style lighting”); (3) subconcept, indicating that the R-concept is a subconcept of the F-concept (e.g., “exterior wall” and “wall”); and (4) associated concept, indicating that the R-concept and the F-concept are associated (bidirectional relationship) (e.g., “floor joist” and “beam”).

Table 5.2 Types of Relationships Considered

Relationship type	Relationship interpretation <sup>1</sup>
Equivalent concept	R is equivalent to F
Subconcept	R is subconcept of F
Superconcept	R is superconcept of F
Associated concept	R and F are associated

<sup>1</sup> R means R-concept; F means F-concept

### 5.2.3.3 Types of Features

The following initial set of eight features were identified, which includes a mix of syntactic (i.e., related to syntax and grammar) and semantic (i.e., related to context and meaning) features (see Table 5.3): (1) RTermNum: the number of terms in the concept name of the R-concept, whether the concept name is unigram, bigram, or multigram; (2) RTermPOS: the type of POS pattern in the concept name of the R-concept, whether the pattern is “N,” “NN,” or “JN;” (3) RMatchType: the match type of R-concept, in terms of which term in the R-concept name matches a term in the F-concept name, whether it is the “first” or “last” term in the R-concept name; (4) RelMatchType: the match type between R-concept and F-concept, whether it is “term-based” match, “synonym”-based match (i.e., the matched term in the F-concept name is a synonym of the matching term in the R-concept name), “hyponym”-based match, or “hypernym”-based match; (5) FMatchType: the match type of F-concept, in terms of which term in the F-concept name matches the matching term in the R-concept name, whether it is “first,” “middle,” or “last;” (6) FTermNum: the number of terms in the concept name of the F-concept, whether the concept name is unigram, bigram, or multigram; (7) FTermPOS: the type of POS pattern in the concept name of the F-concept, whether the pattern is “N,” “NN,” or “JN;” and (8) DOM: the degree of match, which is represented as a Boolean value describing if the R-concept and the F-concept match term by term, with stemming applied, where one represents match and zero represents no match. These features were identified based on the following heuristic rules:

- H4 (see above).
- H6: The type of POS pattern in the name of a concept affects its meaning; and since the concept names are all noun phrases, the most distinguishing POS pattern is whether the concept has a modifier(s), and if yes, whether the modifier(s) is/are nominal (i.e., noun or noun sequences).
- H7: The match type, in terms of which term in each concept name is matched, affects the relationship between the matched concepts.
- H8: The match type, in terms of the type of relationship between the matched terms in both concepts, affects the relationship between the matched concepts.
- H9: If, in the same domain, two concept names match term by term (with stemming applied), then the two concepts are likely to be equivalent.

Table 5.4 shows some example concept pairs and their features. The final set of features is determined after conducting feature selection (as further discussed in the Experimental Testing and Results section).

Table 5.3 The Syntactic and Semantic Features used for the Relationship Classifier

	Feature							
	RTermNum	RTerm POS	RMatch Type	RelMatch Type	FMatch Type	FTermNum	FTerm POS	DOM
Possible values	Unigram, bigram, multigram	N, NN, JN	First, last	Synonym, hypernym, hyponym, term-based	First, middle, last	Unigram, bigram, multigram	N, NN, JN	1, 0

Table 5.4 Example R-Concepts, Matched F-Concepts, and their Feature Values

Concept pair		Feature and feature values							
R-concept	F-concept	RTermNum	RTerm POS	RMatch Type	RelMatchType	FMatch Type	FTermNum	FTerm POS	DOM
Door	Door	Unigram	N	First	Term-based	First	Unigram	N	1
Exterior wall	Wall	Bigram	NN	Last	Term-based	First	Unigram	N	0
Lighting	Surface style lighting	Unigram	N	First	Term-based	Last	Multigram	NN	0
Skylight	Window	Unigram	N	First	Synonym	First	Unigram	N	0
Floor joist	Beam	Bigram	N	Last	Synonym	First	Unigram	NN	0
Water-proof joint	Structural connection	Bigram	NN	Last	Hypernym	Last	Bigram	JN	0

## 5.2.4 Regulatory Concept Integration

### 5.2.4.1 Proposed Concept Integration Approach

The regulatory concept integration phase aims to integrate the extracted and matched regulatory concepts (R-concepts) into the IFC schema. For the R-concepts that are successfully matched with F-concepts, if the relationship between an R-concept and an F-concept was classified as a superconcept or subconcept, then the R-concept is automatically integrated into the IFC schema using mapping rules. If the relationship between an R-concept and an F-concept was classified as an equivalent concept, then no entity for the R-concept is created and, instead, a rule that indicates this equivalent relationship is created (e.g., a logic rule where F-concept forms the body and R-concept forms the head) to process instances of the equivalent R-concepts to F-concepts when needed. If the relationship between an R-concept and an F-concept was classified as associated concept, then an entity for the R-concept is created as a subentity of a newly created concept called “IfcAccConcept,”

instances of the association relations are created using the newly added entities for representing regulatory relations described in Section 5.3.4.1. For the regulatory concepts that are not matched with any IFC concepts, the corresponding entities of these R-concepts are created as subentities of the newly created concept entity “IfcAccConcept.”

#### 5.2.4.2 Mapping Rules

The mapping rules for superconcept and subconcept relationships are shown in Figure 5.6. If an R-concept is a subtype of an F-concept, then an entity for the R-concept is created with the declaration that it is a subtype of the F-concept. If an R-concept is a supertype of an F-concept, then an entity for the R-concept is created with the declaration that it is a supertype of the F-concept.

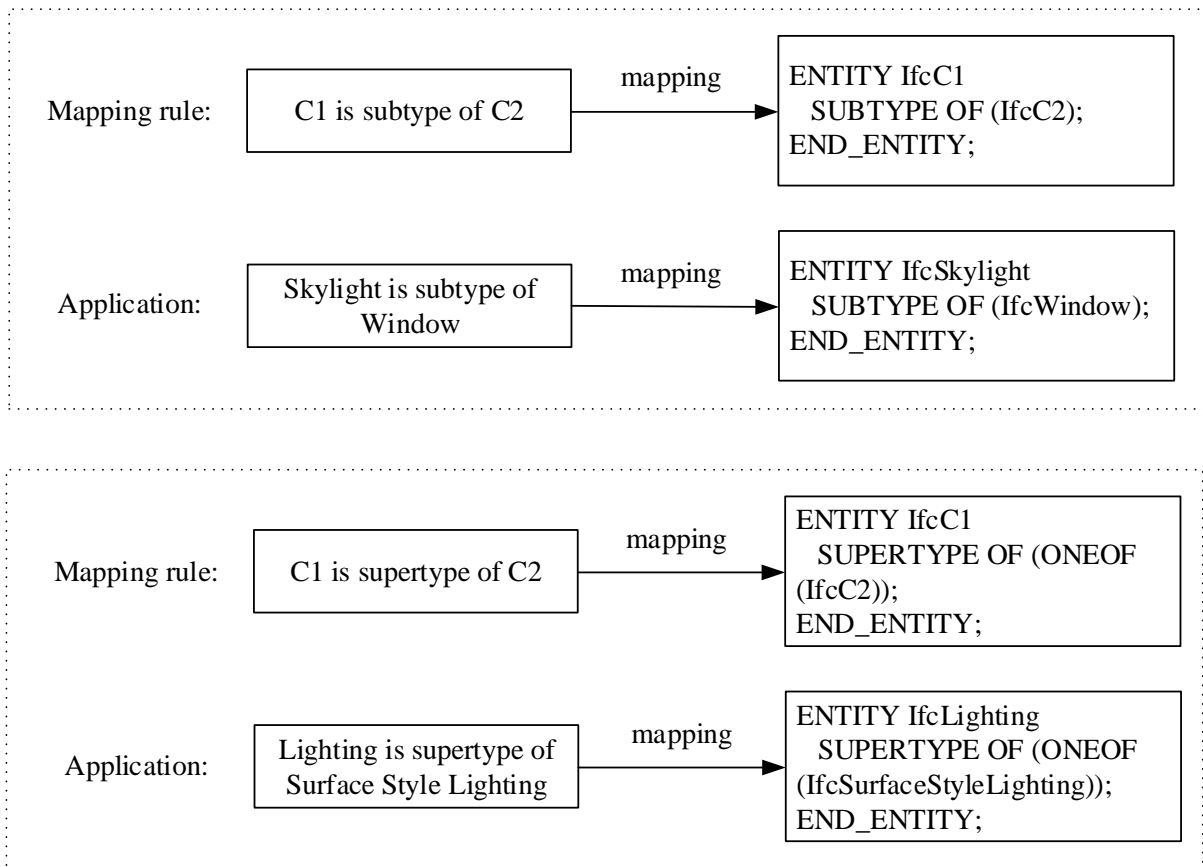


Figure 5.6 Mapping Rules for Regulatory Concept Integration into IFC

### 5.3 Experimental Testing and Evaluation

The proposed method for semiautomated IFC extension was tested on extending the IFC class hierarchy (based on schema version IFC2X3\_TC1) using regulatory concepts from IBC. Two chapters, Chapter 12 of IBC 2006 and Chapter 19 of IBC 2009, were randomly selected. Chapter 12 was used for: (1) developing the set of POS patterns for use in regulatory concept extraction (Phase 1), (2) selecting the best combination of SS scoring function and SS measure for IFC concept selection (Phase 2), and (3) training the ML classifier for relationship classification (Phase 3). Chapter 19 was used for testing and evaluating each of



the following sub-methods/algorithms: regulatory concept extraction, IFC concept selection, relationship classification, and regulatory concept integration. Each submethod/algorithm was tested separately.

### **5.3.1 Testing and Evaluation of Regulatory Concept Extraction**

#### **5.3.1.1 Gold Standard**

The gold standards of R-concepts for Chapter 12 of IBC 2006 and Chapter 19 of IBC 2009 were developed manually by the author. A gold standard refers to a benchmark against which testing results are compared for evaluation. An R-concept is a concept in a regulatory document that defines a “thing” (e.g., subject, object, abstract concept). The longest span for each noun phrase was manually recognized and extracted as an R-concept. The longest span could be multi-term (e.g., “minimum net glazed area”) or single-term (e.g., “window”). For example, concepts in the list *LI* were recognized and extracted from Sentence *S6*. The gold standards of Chapter 12 and Chapter 19 include 368 and 821 concepts, respectively. The concepts extracted by the algorithm are then compared to the concepts in the gold standard for evaluating the algorithm in terms of precision and recall of extracted concepts.

- *S6*: “Wall segments with a horizontal length-to-thickness ratio less than 2.5 shall be designed as columns.”
- *LI*: [‘wall\_segments’, ‘horizontal\_length-to-thickness\_ratio’, ‘columns’]

### 5.3.1.2 Algorithm Implementation

The proposed regulatory concept extraction method was implemented in Python programming language (Python v.2.7.3). The Stanford Parser (version 3.4) (Toutanova et al. 2003) was selected and used to generate the POS tags for each word. The Stanford Parser used Penn Treebank tag set which includes 36 tags. Ten, out of the 36 tags, were used (shown in Figure 5.2).

### 5.3.1.3 Evaluation Metrics

Regulatory concept extraction was evaluated in terms of precision, recall, and F1-measure. Precision, here, is defined as the ratio between the number of correctly extracted concepts and the total number of extracted concepts. Recall, here, is defined as the ratio between the number of correctly extracted concepts and the total number of concepts that should be extracted (i.e., the number of concepts in the gold standard). F1-measure is defined as the harmonic mean of precision and recall. A higher recall is more important than precision because the overall method of IFC extension is semiautomated; precision errors could be detected and eliminated by the user during user concept selection.

### 5.3.1.4 Development Results and Analysis

The development of the set of POS patterns to use in the CE rules was conducted following the algorithm shown in Figure 5.3. Figure 5.7 shows the final set of POS patterns, which consists of 39 patterns. These 39 POS patterns were used as conditions for 39 CE rules, one

POS pattern for one CE rule. For example, the pattern “JJ” “JJ” “JJ” “NN” was used for a CE rule which extracts three consecutive adjectives followed by a singular/mass noun as a concept, such as in the concept “minimum net glazed area.”

Table 5.5 shows the performance of extracting R-concepts from the development text (Chapter 12 of IBC 2006). Through error analysis two sources of errors were found: (1) POS tagging error, which accounted for 38.1% of the errors. For example, “herein” was incorrectly tagged as “NN” instead of the correct tag “RB,” and was thus incorrectly extracted; and (2) ambiguity of the POS tag “VBG” between gerund and present participle, which accounted for 61.9% of the errors. For example, “being” is a present participle and thus does not represent a concept, but it was extracted because the POS tag “VBG” was included in the POS patterns for representing gerund. While addressing the first source of errors depends on the improvement of existing POS taggers, the second source was addressed by adding the false positive present participle terms (e.g., “having,” “being,” “involving”) to the exclusion word list. Membership in the exclusion word list prevents a word/phrase from being extracted in spite of matching a POS pattern in the set. The performance of regulatory concept extraction using the exclusion word list is shown in Table 5.5. Precision increased, from 93.4% to 97.1%, without decreasing recall.

“NN”	“VBN” “NN”	“VBN” “NN” “NN”
“NNP”	“VBN” “NNS”	“VBN” “NN” “NNS”
NNS	“JJ” “JJ” “NN”	“JJ” “JJ” “JJ” “NN”
“VBG”	“JJ” “JJ” “NNS”	“JJ” “JJ” “NN” “NN”
“JJ” “NN”	“JJ” “NN” “NN”	“JJ” “NN” “NN” “NN”
“JJS” “NN”	“JJ” “NN” “NNS”	“NN” “NN” “NN” “NNS”
“JJ” “NNS”	“NN” “NN” “NN”	“NNP” “NN” “NN” “NN”
“NN” “NN”	“NN” “NN” “NNS”	“NNP” “NN” “NN” “NNS”
“NN” “NNS”	“NNP” “NN” “NN”	“NNP” “NNP” “NNP” “NNP”
“NNP” “NNP”	“NNP” “NNP” “NNP”	“RB” “VBN” “JJ” “NN”
“NNP” “NNS”	“NNP” “VBD” “NNS”	“RB” “VBN” “NN” “NN”
“VBG” “NN”	“NN” “VBG” “NN”	“RB” “VBN” “NNP” “NN”
“VBG” “NNS”	“RB” “JJ” “NNS”	“JJ” “VBN” “JJ” “JJ” “VBG” “NN”

Figure 5.7 The Set of Flattened POS Patterns Developed

Table 5.5 Performance of Extracting R-Concepts from Development Text (Chapter 12 of IBC 2006)

Method	Number of R-Concepts in gold standard	Number of extracted R-concepts	Number of correctly extracted R-concepts	Precision	Recall	F1-measure
Without exclusion word list	368	391	365	93.4%	99.2%	96.2%
With exclusion word list	368	376	365	97.1%	99.2%	98.1%

### 5.3.1.5 Testing Results and Discussion

The regulatory concept extraction algorithm was tested on Chapter 19 of IBC 2009. The precision, recall, and F1-measure are 89.4%, 94.2%, and 91.7%, and 88.7%, 94.2%, and 91.4%, with and without the use of exclusion word list, respectively. Table 5.6 shows the

performance results. Through error analysis, when using the exclusion word list, four sources of errors were found: (1) POS tagging errors, which accounted for 20.7% of the errors. For example, “corresponding” was incorrectly tagged as “NN” (as opposed to “VBG”); and, thus, “force\_level\_corresponding” was incorrectly extracted as a concept; (2) ambiguity of POS tag “VBG” between gerund and present participle, which accounted for 8.7% of the errors. For example, “excluding” was incorrectly extracted as a concept because the POS tag for present participle was “VBG” (although it does not represent a meaningful nominal concept); (3) word continuation using hyphen, which accounted for 27.2% of the errors. For example, “pro\_vide” was incorrectly extracted as a concept because the word continuation in “pro-vidē” led to “pro” and “vide” be tagged as two words with the tags “JJ” and “NN;” and (4) missing POS patterns, which accounted for 43.5% of the errors. For example, “concrete\_breakout\_strength” and “breakout\_strength\_requirements” were incorrectly extracted as two concepts (instead of one concept, “concrete\_breakout\_strength\_requirements”) because the POS pattern “JJ” “JJ” “JJ” “NN” “NNS” was missing.

Preventing errors from source (1) requires improvement of POS taggers. Preventing errors from source (3) requires a better word continuation representation manner instead of using hyphen, in order to avoid confusion with hyphens used for conjoining noun modifiers. Preventing errors from sources (2) and (4) could be partially prevented by further developing the exclusion word list and POS pattern set, respectively. The use of the developed exclusion

word list [to prevent errors from source (2)] prevented 6 instances of false positives and increased precision from 88.7% to 89.4%. More terms could be added, iteratively, to the exclusion word list to further enhance performance. Similarly, errors from source (4) could be prevented by adding more patterns to the POS pattern set until all possible POS patterns are included. While theoretically this POS pattern set is infinite (e.g., infinite number of “JJ” before a “NN”), in practice this POS pattern set is quite limited [e.g., words with more than 7 prenominal modifiers (e.g., white thin high strong stone north exterior ancient wall) are seldom (if not never) seen].

To test the effect of iterative development of the exclusion word list and POS pattern set, three more experiments were conducted to: (1) add the false positive present participle terms (identified as a result of initial testing) to the exclusion word list and use it in further testing; (2) add the missing POS patterns (identified as a result of initial testing) to the pattern set and use it in further testing; and (3) use both, the extended exclusion word list and the extended POS pattern set, in further testing. Table 5.7 shows the performance results of the three experiments. The results show that the use of the extended exclusion word list and the POS pattern set both improve the performance of concept extraction, with the latter showing a larger improvement.

Table 5.6 Performance of Extracting R-Concepts from Testing Text (Chapter 19 of IBC 2009)

Method	Number of R-concepts in gold standard	Number of extracted R-concepts	Number of correctly extracted R-concepts	Precision	Recall	F1-measure
Without exclusion word list	821	871	773	88.7%	94.2%	91.4%
With exclusion word list	821	865	773	89.4%	94.2%	91.7%

Table 5.7 Performance of Regulatory Concept Extraction after Improvements

Method	Number of R-concepts in gold standard	Number of extracted R-concepts	Number of correctly extracted R-concepts	Precision	Recall	F1-measure
Baseline condition (from Table 5.6)	821	865	773	89.4%	94.2%	91.7%
With extended exclusion word list	821	856	774	90.4%	94.3%	92.3%
With extended POS pattern set	821	860	784	91.2%	95.5%	93.3%
With both extended exclusion word list and extended POS pattern set	821	851	785	92.2%	95.6%	94.0%

### 5.3.2 Testing and Evaluation of IFC Concept Selection

#### 5.3.2.1 Gold Standard

The gold standards of F-concepts for Chapter 12 of IBC 2006 and Chapter 19 of IBC 2009 were developed manually by the author. The F-concepts were initially identified using the matching and ranking algorithms and then filtered manually. The gold standards of Chapter 12 and Chapter 19 include 343 and 588 F-concepts, respectively.

#### 5.3.2.2 Algorithm Implementation

The proposed IFC concept selection method and algorithm were implemented in Python programming language (Python v.2.7.3). The Porter Stemmer (Porter 1980) was used for

stemming. The “re” (regular expression) module in python was utilized to support the matching algorithms. The hypernymy, hyponymy, and synonymy relations in WordNet were utilized through the Natural Language Toolkit (NLTK) (Bird et al. 2009) WordNet interface in python.

### 5.3.2.3 Evaluation Metrics

IFC concept selection was evaluated in terms of adoption rate. Adoption rate, here, is defined as the number of automatically selected F-concepts that were adopted divided by the total number of automatically selected F-concepts.

### 5.3.2.4 Development Results and Analysis

For term-based matched F-concepts, Table 5.8 shows the results of testing combinations of different concept-level SS scoring functions and term-level SS scoring functions. Table 5.9 shows some example concepts that were extracted and matched using the different combinations. For concept-level SS scoring, Equation (5.1) and Equation (5.2) were tested. As shown in Table 5.8, Equation (5.1) consistently outperformed Equation (5.2). Equation (5.1) prefers shorter F-concepts and, thus, tends to select F-concepts that are higher in the concept hierarchy (most likely a superclass). In comparison, Equation (5.2) prefers F-concepts with similar length to the R-concept and, thus, tends to select F-concepts that are at a similar level in the concept hierarchy to the R-concept. However, an F-concept located at a similar level to the R-concept may deviate a lot in meaning because concepts at similar level in a concept hierarchy could belong to different branches of the hierarchy. A matched



higher-level F-concept, thus, usually has higher relatedness to the R-concept than a matched similar-level F-concept. For example, using Shortest Path Similarity (for term-level SS scoring), Equation (5.1) resulted in matching of “net\_free\_ventilating\_area” and “quantity\_area,” whereas Equation (5.2) resulted in the matching of “net\_free\_ventilating\_area” and “annotation\_fill\_area\_occurrence.” “Quantity\_area” was correctly a superconcept of “net\_free\_ventilating\_area” and was adopted. On the other hand, the meaning of “annotation\_fill\_area\_occurrence” was far from that of “net\_free\_ventilating\_area” despite being at a similar level in the concept hierarchy. Based on these experimental results, Equation (5.1) was selected for concept-level SS scoring for term-based matched F-concepts.

For term-level SS scoring, the following five existing SS measures were tested: Shortest Path Similarity, Jiang-Conrath Similarity, Leacock-Chodorow Similarity, Resnik Similarity, and Lin Similarity (see Section 2.5.3). The Shortest Path Similarity is the simplest among the five tested measures, and achieved the best adoption rate of 87.1%. The Shortest Path Similarity and Leacock-Chodorow Similarity are based on shortest path between two concepts in a taxonomy. The other three SS measures are based on information content of the two concepts’ least common subsumer (i.e., the lowest-level concept that is a superconcept of both concepts). The performance drop from the Shortest Path Similarity to the other similarity measures (except for Leacock\_Chodorow Similarity) shows the advantage of a shortest path measure in comparison to an information content of the least common subsumer measure.

Empirically, this is because the length of path between two concepts is more distinctive than the information content of their least common subsumer. For shortest path measures, the Leacock-Chodorow Similarity takes the depth of the taxonomy into consideration, in addition to the use of shortest path. The performance drop from the Shortest Path Similarity to the Leacock-Chodorow Similarity indicates that the absolute taxonomy depth is not a distinctive feature in the context of concept matching. Based on these experimental results, the Shortest Path Similarity was selected for term-level SS scoring for term-based matched F-concepts.

Table 5.8 Performances of Different SS Scoring Methods for Term-Based Matched F-Concepts

Proposed concept-level SS scoring function	Term-level SS scoring function	Number of related F-concepts found	Number of related F-concepts adopted	Adoption rate
Eq. (5.1)	Shortest Path Similarity	286	249	87.1%
Eq. (5.2)	Shortest Path Similarity	286	225	78.7%
Eq. (5.1)	Jiang-Conrath Similarity	286	244	85.3%
Eq. (5.2)	Jiang-Conrath Similarity	286	224	78.3%
Eq. (5.1)	Leacock-Chodorow Similarity	286	237	82.9%
Eq. (5.2)	Leacock-Chodorow Similarity	286	202	70.6%
Eq. (5.1)	Resnik Similarity	286	246	86.0%
Eq. (5.2)	Resnik Similarity	286	228	79.7%
Eq. (5.1)	Lin Similarity	286	246	86.0%
Eq. (5.2)	Lin Similarity	286	224	78.3%

Table 5.9 Examples of Matched R-Concepts and F-Concepts Using Different SS Scoring Methods for Term-Based Matched F-Concepts

Extracted R-concept	Proposed concept-level SS scoring function	Matched F-concept using Shortest Path Similarity <sup>1</sup>	Matched F-concept using Leacock-Chodorow Similarity <sup>1</sup>	Matched F-concept using Jiang-Conrath Similarity <sup>1</sup>
Exterior wall	Eq. (5.1)	Wall	Wall	Wall
	Eq. (5.2)	Curtain wall	Curtain wall	Curtain wall
Lighting	Eq. (5.1)	Surface style lighting	Light source	Surface style lighting
	Eq. (5.2)	Surface style lighting	Light source	Surface style lighting
Conditioned space	Eq. (5.1)	Space	Space	Space
	Eq. (5.2)	Interior space	<i>Space program</i>	Interior space
Dwelling unit entrance door	Eq. (5.1)	Door	Door	Door
	Eq. (5.2)	Door	<i>Door lining properties</i>	Door
Net free ventilating area	Eq. (5.1)	Quantity area	Quantity area	Quantity area
	Eq. (5.2)	<i>Annotation fill area occurrence</i>	<i>Annotation fill area occurrence</i>	<i>Annotation fill area occurrence</i>

<sup>1</sup> italicized concepts were not adopted

For semantic-based matched F-concepts, Table 5.10 shows the results of testing combinations of different concept-level SS scoring functions and term-level SS scoring functions. Table 5.11 shows some examples of concepts that were extracted and matched using the different combinations. As shown in Table 5.10, for concept-level SS scoring, Equation (5.1) and Equation (5.2) did not show any variability in performance. Since both functions performed equally, for consistency with term-based matching of F-concepts, Equation (5.1) was selected for concept-level SS scoring for semantic-based matched F-concepts.

For term-level SS scoring, the Shortest Path Similarity outperformed all other SS measures. This is consistent with the results obtained for term-based matched F-concepts. Based on the experimental results, the Shortest Path Similarity was selected for term-level SS scoring for

semantic-based matched F-concepts.

Thus, the same term-level SS scoring function (Shortest Path Similarity) and concept-level SS scoring function [Equation (5.1)] were selected for both term-based matching and semantic-based matching algorithms. This shows consistency of performance across both types of matching.

Table 5.10 Performances of Different SS Scoring Methods for Semantic-Based Matched F-Concepts

Proposed concept-level SS scoring function	Term-level SS scoring function	Number of related F-concepts found	Number of related F-concepts adopted	Adoption rate
Eq. (5.1)	Shortest Path Similarity	114	94	<b>82.5%</b>
Eq. (5.2)	Shortest Path Similarity	114	94	<b>82.5%</b>
Eq. (5.1)	Jiang-Conrath Similarity	114	92	80.7%
Eq. (5.2)	Jiang-Conrath Similarity	114	92	80.7%
Eq. (5.1)	Leacock-Chodorow Similarity	114	93	81.6%
Eq. (5.2)	Leacock-Chodorow Similarity	114	93	81.6%
Eq. (5.1)	Resnik Similarity	114	93	81.6%
Eq. (5.2)	Resnik Similarity	114	93	81.6%
Eq. (5.1)	Lin Similarity	114	93	81.6%
Eq. (5.2)	Lin Similarity	114	93	81.6%

Table 5.11 Examples of Matched R-Concepts and F-Concepts Using Different SS Scoring Methods for Semantic-Based Matched F-Concepts

Extracted R-concept	Proposed concept-level SS scoring Function	Matched F-concept using Shortest Path Similarity <sup>1</sup>	Matched F-Concept using Leacock-Chodorow Similarity <sup>1</sup>	Matched F-Concept using Jiang-Conrath Similarity <sup>1</sup>
corrosion-resistant	Eq. (5.1)	hardware cloth	hardware cloth	hardware cloth
wire cloth screening	Eq. (5.2)	hardware cloth	hardware cloth	hardware cloth
squirrels	Eq. (5.1)	rodents	rodents	rodents
	Eq. (5.2)	rodents	rodents	rodents
outdoors	Eq. (5.1)	outside horizontal clear space	outside horizontal clear space	outside horizontal clear space
	Eq. (5.2)	outside horizontal clear space	outside horizontal clear space	outside horizontal clear space
installed shower heads	Eq. (5.1)	<i>contaminant sources</i>	<i>contaminant source</i>	<i>light source</i>
	Eq. (5.2)	<i>contaminant sources</i>	<i>contaminant source</i>	<i>light source</i>

<sup>1</sup> italicized concepts were not adopted

### 5.3.2.5 Testing Results and Discussion

The proposed IFC concept selection method and algorithm [using Equation (5.1) and Shortest Path Similarity] were tested in automatically selecting F-concepts for the extracted R-concepts (from Phase I). The testing results are summarized in Table 5.12. The total adoption rate is 84.5%. The adoption rates for term-based and semantic-based matched F-concepts are 84.8% and 82.7%, respectively, both which are close to the training performance (87.1% and 82.5%, respectively). This shows initial stability in the performance of the proposed IFC concept selection method.

Table 5.12 Testing Results of IFC Concept Selection Method

Concept matching type	Concept-level SS scoring function	Term-level SS scoring function	Number of related F-concepts found	Number of related F-concepts adopted	Adoption rate
Term-based matching	Eq. (5.1)	Shortest Path Similarity	598	507	84.8%
Semantic-based matching			98	81	82.7%
Total			696	588	84.5%

### 5.3.3 Testing and Evaluation of Relationship Classification

#### 5.3.3.1 Gold Standard

The aim of the classifier is to predict the relationship between each pair of R-concept and F-concept. Two gold standards, one for training and one for testing, were developed manually by the author and verified by three other researchers. The training and testing gold standards included pairs of concepts from Chapter 12 of IBC 2006 and Chapter 19 of IBC 2009, respectively. The training data set was used for feature selection, ML algorithm selection, and classifier training. The testing data set was used for evaluating the classifier’s performance. In each gold standard, the relationship between each R-concept and F-concept was defined. Four types of relationships were defined, as per Table 5.2. Table 5.13 shows some example concept pairs and their corresponding relationships.

Table 5.13 Examples of Matched R-Concepts and F-Concepts and Corresponding Relationships

Concept pair		Relationship
Extracted R-concept	Matched F-concept	
Diameter	Diameter dimension	Equivalent
Skylight	Window	Subconcept
Lighting	Surface style lighting	Superconcept
Water-proof joint	Structural connection	Associated concept

### 5.3.3.2 Algorithm Implementation

The proposed relationship classification algorithm was developed and tested in the Waikato Environment for Knowledge Analysis (Weka) data mining software system (Hall et al. 2009). A program for generating the ML features was developed using Python programming language (Python v.2.7.3). The following ML algorithms were tested: (1) `weka.classifiers.bayes.NaiveBayes` for Naïve Bayes; (2) `weka.classifiers.trees.J48` for Decision Tree (DT); (3) `weka.classifiers.lazy.IBk` for k-NN; and (4) `weka.classifiers.functions.SMO` for SVM. Tenfold cross-validation was applied to each training experiment, which randomly split the data to a training subset and a testing subset ten times and averaged the results from the ten rounds of training and testing.

### 5.3.3.3 Evaluation Metrics

Relationship classification was evaluated in two ways: (1) the performance across all relationships was evaluated, together, in terms of precision, and (2) the performance for each type of relationship was evaluated, separately, in terms of precision, recall, and F1-measure. In the first case, precision is defined as the number of correctly classified concept pairs divided by the total number of classified concept pairs. In the second case, precision is defined as the number of correctly classified concept pairs in a relationship type divided by the total number of concept pairs that are classified into that relationship type. Recall is defined as the correctly classified concept pairs in a relationship type divided by the total number of concept pairs that should be classified into that relationship type. F1-measure is

the harmonic mean of precision and recall.

#### 5.3.3.4 ML Algorithm Selection, Feature Selection, and Classifier Training

The training data set was used for feature selection and classifier training. The results of testing the four ML algorithms, prior to feature selection, are summarized in Table 5.14. While three out of the four ML algorithms achieved a precision greater than 85%, k-NN achieved the best precision of 90.98% (using the Polynomial kernel) followed by SVM with a precision of 90.71%.

A “leave-one-out” feature analysis was used for feature selection. Feature selection, in this dissertation, aims to select, based on performance, a subset (or the full set) of the complete/initial feature set (the eight features, see Table 5.3) for use in representing the concepts. The “leave-one-out” feature analysis is a method to analyze the contribution of each feature by comparing the performances with and without that feature. The analysis was conducted using the top-three performing ML algorithms (k-NN, SVM, and DT). The feature analysis results are summarized in Table 5.15. The bold highlighted values indicate the precision values that outperformed the baseline precision (where all eight features were used). The results show that four out of the eight features (RTermNum, RTermPOS, RelMatchType, FTermNum) were not discriminating when using DT, one out of the eight features (FTermNum) was not discriminating when using k-NN, and all eight features were discriminating when using SVM. Using only the discriminating features (i.e., RMatchType, FMatchType, FTermPOS, and DOM for DT, RTermNum, RTermPOS, RMatchType,



RelMatchType, FMatchType, FTermPOS, and DOM for k-NN, and all eight features for SVM), DT achieved a precision of 87.43%, k-NN achieved a precision of 91.26%, and SVM achieved a precision of 90.71%. This difference shows that, in comparison to DT, k-NN and SVM were able to achieve higher performances with larger feature sizes. This may indicate that the additional features used by k-NN and SVM provided better discriminating ability to the classifiers. As such, based on the experimental results, the aforementioned seven discriminating features and the k-NN algorithm were selected for training the classifier.

The results also show that the following four features were discriminating for all three algorithms: RMatchType, FMatchType, FTermPOS, and DOM. DOM was discriminating because a term-by-term match could provide a strong indication of concept equivalency. The fact that RMatchType and FMatchType were discriminating shows that the arrangement of terms could affect the meanings of concepts and that the locations of the matching terms in a concept pair could affect the relationship between the two concepts in the pair. In addition to these four features, the following three features were discriminating for k-NN and SVM: RTermNum, RTermPOS, and RelMatchType. The fact that these features were discriminating in k-NN and SVM but not in DT may be attributed to the different types of ML algorithms. More importantly, the fact that the RelMatchType is discriminating shows that the semantic features could benefit the task of concept relationship classification and result in further improvement of precision.

Table 5.14 Results of Testing Different Machine Learning Algorithms (Prior to Feature Selection)

Metric	Machine learning algorithm			
	K-NN	SVM	Decision Tree	Naïve Bayes
Total number of relationship instances	366	366	366	366
Number of correctly classified relationship instances	333	332	315	279
Precision	90.98%	90.71%	86.07%	76.23%

Table 5.15 Leave-One-Out Feature Analysis Precision Results

ML algorithm	Precision result when feature excluded								
	None	RTermNum	RTermPOS	RMatchType	RelMatchType	FMatchType	FTermNum	FTermPOS	DOM
k-NN	90.98%	89.07%	89.89%	86.34%	88.80%	86.89%	<b>91.26%</b>	90.44%	88.25%
SVM	90.71%	89.34%	89.62%	87.43%	88.25%	86.89%	90.44%	90.44%	87.43%
Decision Tree	86.07%	<b>86.89%</b>	<b>86.61%</b>	81.15%	<b>86.61%</b>	84.70%	<b>86.89%</b>	86.07%	81.98%

Note: bolded precision results are higher than the baseline precision (none of the features excluded)

### 5.3.3.5 Testing Results and Discussion

The testing data set was used for testing and evaluating the performance of the classifier. The testing results are summarized in Table 5.16. The overall precision across all relationships is 87.94%. This is close to the overall training precision (91.26%), which shows the initial stability in the performance of the relationship classifier. The subconcept relationship type achieved the best precision of 93.4% and the best recall of 93.4%. The analysis of the results shows that in many cases, the R-concept was a bigram or multigram (e.g., “structural concrete”) whose last term matched with the only term in a unigram F-concept (e.g., “concrete”). This pattern has a strong predictive effect. Comparing to the subconcept relationship type, the superconcept relationship type shares a similar pattern but did not achieve a performance as high. The precision and recall for the superconcept relationship

type were 88.5% and 75.4%, respectively. One observation was that the classifier tends to prefer subconcept relationship types over superconcept relationship types, when both the R-concept and the F-concept were bigram or multigram. For example, there were six cases where a superconcept relationship was incorrectly classified as a subconcept relationship, but zero cases where a subconcept relationship was incorrectly classified as a superconcept relationship. This could be due to the fact that there were only two instances of bigram/multigram concept pairs with superconcept relationship in the training data set. The equivalent relationship type achieved a precision of 91.9% and recall of 86.1%. The associated relationship type achieved a precision of 62.5% and recall of 80.0%, which is the lowest among the four types of relationships. This is probably because: (1) the size of the training data was limited for this relationship type, and (2) the associated relationship includes more semantic types than the other types of relationships and has more variability in the expression of concepts. Thus, while the data set might provide enough variability for concepts related to the other relationship types, the associated relationship may require more data. Overall, the precision is 87.94%, which is considered a good performance [within the range of 80% to 90% (Spiliopoulos et al. 2010)].

Table 5.16 Relationship Classifier Testing Results

Relationship type	Number of relationship instances in gold standard	Number of classified relationship instances	Number of correctly classified relationship instances	Precision	Recall	F1-Measure
Equivalent concept	79	74	68	91.9%	86.1%	88.9%
Subconcept	241	241	225	93.4%	93.4%	93.4%
Superconcept	61	52	46	88.5%	75.4%	81.4%
Associated concept	50	64	40	62.5%	80.0%	70.2%
Total	431	431	379	87.94%	87.94%	87.94%

### 5.3.4 Testing and Evaluation of Regulatory Concept Integration

#### 5.3.4.1 Gold Standard

The gold standard of an extended IFC schema was developed by manually adding regulatory concepts in Chapter 19 of IBC 2009 to the IFC\_2X3\_TC1 schema (BuildingSmart 2014). The regulatory concepts were added based on their relationship with the IFC concepts as described in Section 5.2.4. As a result, in addition to the original concept entities from the IFC\_2X3\_TC1 schema, the extended IFC schema includes 743 concept entities of regulatory concepts from Chapter 19 of IBC 2009. These additional concept entities include: (1) A concept entity “IfcAccConcept,” which was added as a subtype of “IfcObject;” (2) 241 concept entities that were added as subtypes of original entities in the IFC\_2X3\_TC1 schema. For example, “IfcThinEdge” was added as a subtype of “IfcEdge;” (3) 61 concept entities that were added as supertypes of original entities in the IFC\_2X3\_TC1 schema. For example, “IfcConnections” was added as a supertype of “IfcStructuralConnection;” (4) 50 concept

entities that were added as direct subtypes of “IfcAccConcept,” which were identified as associated concepts of original entities in the IFC\_2X3\_TC1 schema. For example, “IfcBasement” was added as a subtype of “IfcAccCocnept,” which was identified as an associated concept of “IfcBasementWall;” and (5) 390 concept entities that were added as direct subtypes of “IfcAccConcept” because they were extracted from Chapter 19 of IBC 2009 but were not matched with any original entities in the IFC\_2X3\_TC1 schema. For example, “IfcWallPier” was added as a direct subtype of “IfcAccConcept.” Table 5.17 shows some examples of the additional concept entities in the extended IFC schema.

Table 5.17 Examples of the Additional Concept Entities in the Extended IFC Schema

Added concept entity	Supertype entity in extended schema	Subtype entity in extended schema
IfcAccConcept	IfcObject	-
IfcThinEdge	IfcEdge	-
IfcSkylight	IfcWindow	-
IfcConnections	-	IfcStructuralConnection
IfcLighting	-	IfcSurfaceStyleLighting
IfcBasement	IfcAccConcept	-
IfcWallPier	IfcAccConcept	-

In addition to the original “IfcRel...” entities designated to represent relations from the IFC\_2X3\_TC1 schema, the extended IFC schema includes six new entities for representing regulatory relations in Chapter 19 of IBC 2009: “IfcAccRelation,” “IfcAccUniRelation,” “IfcAccBiRelation,” “IfcAccTriRelation,” “IfcAccHasUniQuantity,” and “IfcAccHasBiQuantity.” The relation entity “IfcAccRelation” was added as a subtype of “IfcRelationship.” “IfcAccUniRelation,” “IfcAccBiRelation,” and “IfcAccTriRelation” were added as subtypes of “IfcAccRelation” for representing relations for one entity, relations

between two entities, and relations among three entities, respectively. “IfcAccHasUniQuantity” and “IfcAccHasBiQuantity” were added as subtypes of “IfcAccRelation” for representing quantitative relations with one quantity reference or one value and unit set and quantitative relations with two quantity references or two values and unit sets, respectively.

#### 5.3.4.2 Algorithm Implementation

The proposed regulatory concept integration method was implemented in Python programming language (Python v.2.7.3). Three main functions were developed and used for integrating regulatory concepts into the IFC schema based on three types of relationships of regulatory concepts with IFC concepts: subconcept, superconcept, and associated concept. The creation of rules for regulatory concepts that are equivalent concepts with IFC concepts were not included in the Python program.

#### 5.3.4.3 Evaluation Metrics

Regulatory concept integration was evaluated in terms of precision, recall, and F1-measure. Precision, here, is defined as the ratio between the number of correctly integrated concepts and the total number of integrated concepts. Recall, here, is defined as the ratio between the number of correctly integrated concepts and the total number of concepts that should be integrated (i.e., the number of concepts in the gold standard). F1-measure is defined as the harmonic mean of precision and recall.

#### 5.3.4.4 Testing Results and Discussion

The proposed regulatory concept integration method and algorithm were tested in automatically integrating the extracted regulatory concepts into the IFC\_2X3\_TC1 schema (BuildingSmart 2014). The testing results are summarized in Table 5.18. As shown in the Table, all regulatory concepts were successfully integrated into the IFC schema which led to a 100% performance for all precision, recall, and F1-measure. This shows initial effectiveness of the proposed regulatory concept integration method.

Table 5.18 Performance of Integrating R-Concepts from Chapter 19 of IBC 2009 into the IFC\_2X3\_TC1 Schema

Relationship Type	Number of integrated R-concepts in gold standard	Number of integrated R-concepts	Number of correctly integrated R-concepts	Precision	Recall	F1-measure
Subconcept	241	241	241	100%	100%	100%
Superconcept	61	61	61	100%	100%	100%
Associated concept	50	50	50	100%	100%	100%
Non	390	390	390	100%	100%	100%
Total	742	742	742	100%	100%	100%

## **6 CHAPTER 6 – AUTOMATED INFORMATION EXTRACTION FROM BUILDING INFORMATION MODELS AND TRANSFORMATION OF DESIGN INFORMATION**

### **6.1 Comparison to the State of the Art**

To conduct ACC, design information needs to be automatically extracted from BIMs into a representation that can be directly used for automated compliance reasoning.

Existing BIM information extraction efforts have taken various different approaches for various purposes. For example, Kim et al. (2013) utilized ifcXML parsers (implemented in Ruby programming language) to extract spatial, quantity, material, and relational information of building elements from IFC-based BIMs, for automatically generating construction schedules. Zhang and Issa (2013) utilized an ontology (implemented in Java programming language) that was coded in web ontology language (OWL) to extract partial models of IFC-based BIMs based on the IFC schema, for reducing the size and complexity of BIMs. There are also existing efforts in extracting information from BIMs to support automated compliance checking. These efforts extract BIM information into different types of representations. For example, Yurchyshyna et al. (2008) and Pauwels et al. (2011) utilized Extensible Stylesheet Language Transformations (XSLT) transformation method to extract information from an IFC-based BIM into a Resource Description Framework (RDF) graph to support regulatory requirement checking in general. Sinha et al. (2013) utilized Revit Application Programming Interface (API) methods to extract building parametric data from



Revit BIMs, for supporting automated compliance checking against energy code criteria. Tan et al. (2010) utilized Java classes to extract wall attributes from IFC-based BIMs, for supporting automated building envelope design checking against building code requirements. Further, the development of the ifcOWL ontology enables the extraction of IFC-based BIM information based on the domain knowledge captured in the ontology, which could further serve the purpose of compliance checking (Beetz et al. 2009; Kadolsky et al. (2014). In addition to these research efforts, commercial BIM software implementations such as ArchiCAD, Autodesk Revit, and Solibri Model Checker have their proprietary methods to access and extract information from IFC-based BIMs.

The BIM extraction methods can be largely categorized into two types: IFC-based and proprietary data format-based. From the IFC-based BIM extraction methods, two subtypes could be defined: XML-based and EXPRESS-based. Both XML and EXPRESS are official schema used for IFC-based BIM data representation. However, because the EXPRESS schema is the default data schema, and BIM data represented using the EXPRESS schema is much smaller (usually 1/3 to 1/4 in size) than the XML-based BIM data (BuildingSmart 2015), extracting information from EXPRESS-based BIMs is slightly preferred than extracting information from XML-based BIMs.

When used for automated compliance checking, BIM extraction efforts are dependent on the source and target representation of the extracted information. The proprietary extraction methods and in-house extraction methods used in commercial softwares cannot be used to

fulfil the IFC-based BIM extraction need in this dissertation because of the differences in representing source and target information. The XSLT transformation method and ifcOWL-based extraction methods, which represent the state-of-the-art IFC-based BIM extraction methods, are more relevant to the scope in this dissertation. The author could have adapted those methods. However, the author decided to take a different approach for the following two reasons: (1) Extraction from EXPRESS-based data is preferred than extraction from XML-based data, and extraction from EXPRESS-based data is feasible because of the existing data access methods for EXPRESS data such as JSDAI; and (2) Extraction into a logic format is preferred than extraction into an ontology, because logic is the final representation used for reasoning.

To enable automated reasoning for supporting ACC, another issue related to BIM extraction is the alignment of design information with regulatory information. For example, in the ACC effort of Yurchyshyna et al. (2008), project information are extracted into an RDF representation. Then, the RDF-represented project information was aligned with the RDF-represented regulatory rules using similarity measures between the two. In the effort of Beetz et al. (2009), an ifcOWL ontology was converted from the EXPRESS schema of IFC, which could be used to support extraction of IFC-based BIM, into an RDF representation. When used in the context of ACC, those RDF representations of building information need to be linked to regulatory information algorithmically or manually (Kadolsky et al. 2014). Comparing to these efforts, the alignment of design and regulatory information is conducted

as an automated internal process in logic, because the representations of both the design information and the regulatory information are in the same logic format (Chapter 7). The domain knowledge in building regulations, when needed to align concepts and relationships, could be directly formalized into logic rules.

In this chapter, a BIM information extraction and transformation method is proposed for automatically extracting all BIM information (i.e., entities and their attributes, excluding detailed geometric representation information) from IFC-based BIMs, and transforming the extracted information into logic facts that could be directly used for logic-based automated reasoning. The proposed method utilizes the Java Standard Data Access Interface (JSDAI) and a set of transformation rules. The chapter presents the details of the proposed method and its testing results using a BIM test case.

## **6.2 Proposed BIM Information Extraction and Transformation Method and Algorithm**

The proposed method for IFC-based BIM information extraction and transformation includes two main phases (as per Figure 6.1): (1) BIM information extraction: extracting BIM information from an .ifc file into a tuple-format, and (2) BIM information transformation: transforming the extracted tuple-represented information into logic facts. Figure 6.2 shows an example of the inputs and outputs of information extraction and transformation.

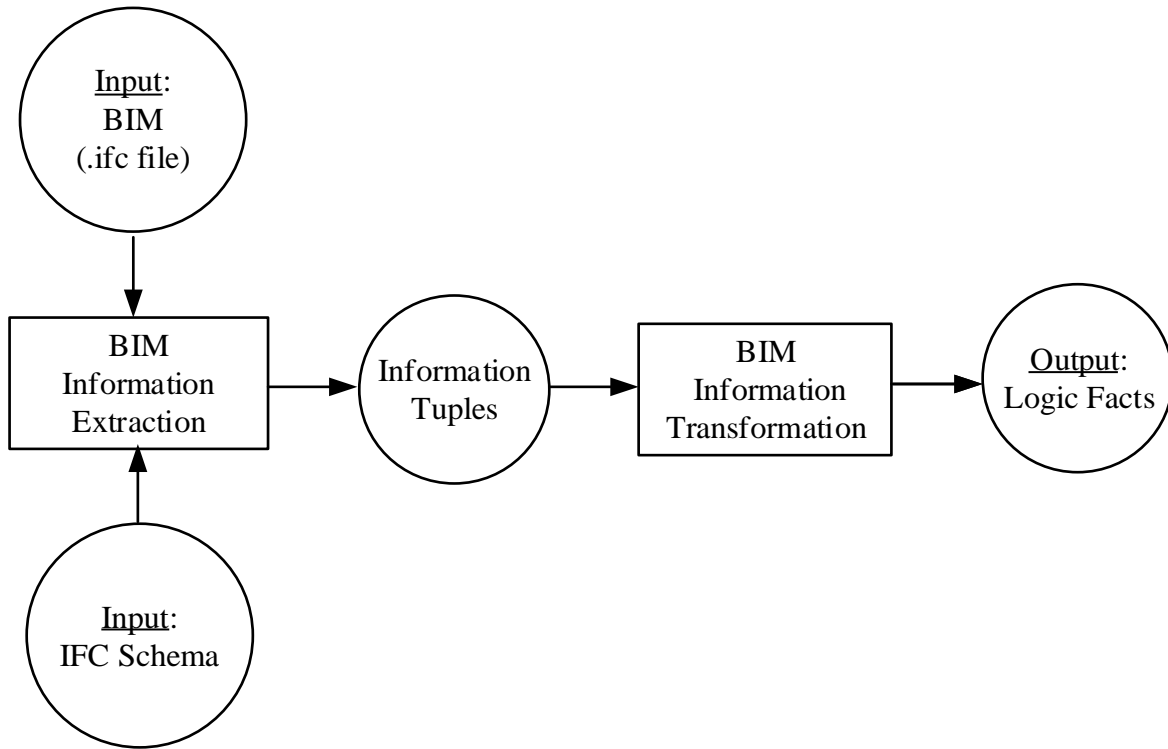


Figure 6.1 The Proposed BIM Information Extraction and Transformation Method

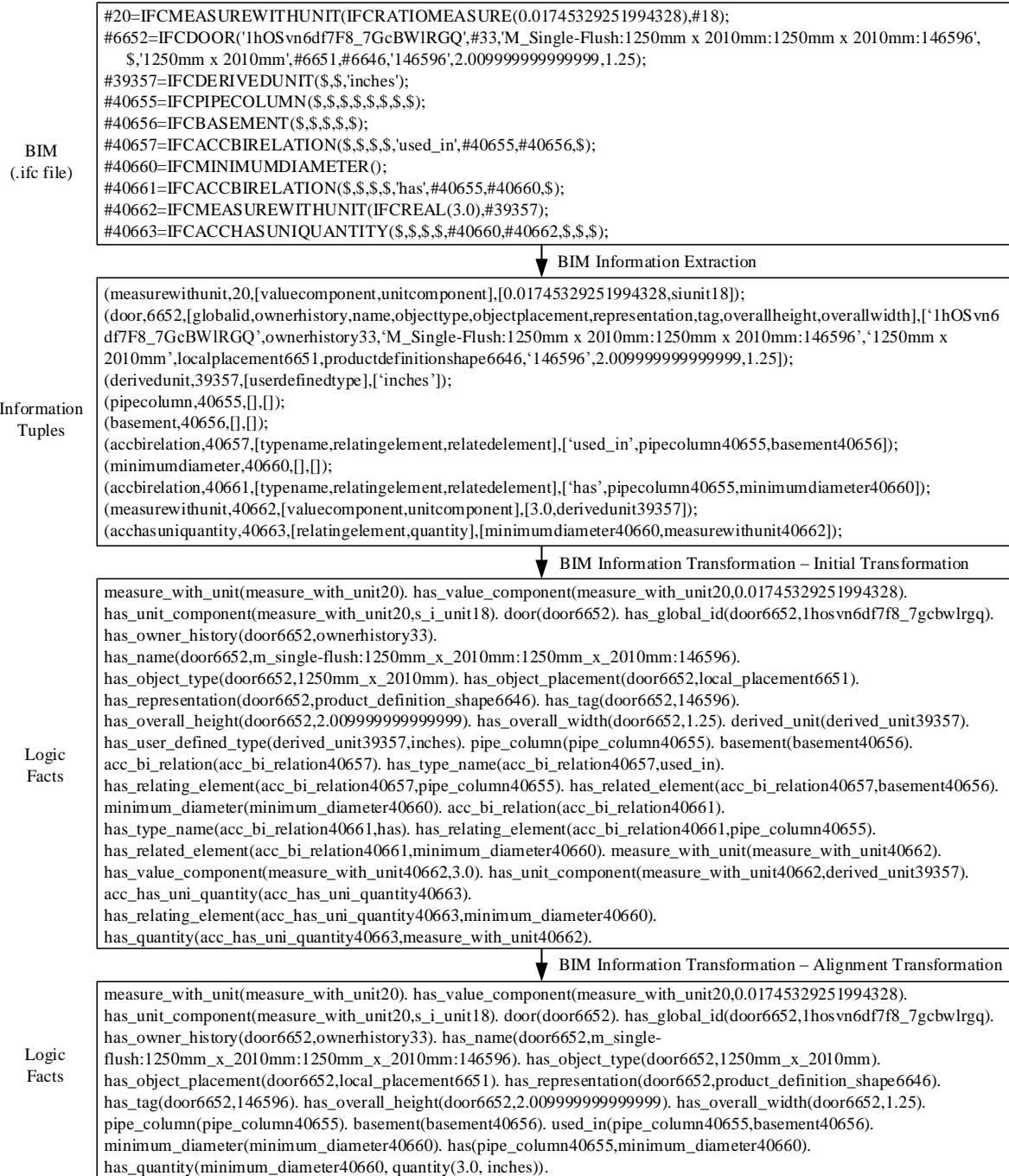


Figure 6.2 Example Illustrating the Inputs and Outputs of BIM Information Extraction and Transformation

## 6.2.1 BIM Information Extraction

### 6.2.1.1 Information Representation

This phase aims to define the representation format of the extracted BIM information. In this

method, the ultimate representation format is logic facts that could be directly used in a logic programming platform for automated reasoning. For intermediate processing, a tuple format is proposed to represent the extracted information. The use of a tuple format for intermediate processing is proposed to facilitate computer manipulation; the simple and clear structure of tuple format facilitates efficient information processing. A four-tuple was used for intermediate information representation: <entity name, entity line ID, attribute name list, attribute value list>. An entity name is the name of an entity. An entity line ID is the line number of the entity in the .ifc file, which is used to identify the entity and distinguish it from other entities. An attribute name list includes the names of the explicit attributes of an entity, where the explicit attributes of an entity include both the explicit attributes in the entity's own definition and the explicit attributes in the definitions of its supertypes. Derived attributes are not processed because they are mostly representing geometric representation details. Inversed attributes are not processed because the relationship identified by these inversed attributes are already represented in explicit attributes. An attribute value list includes the values of the explicit attributes of an entity. Table 6.1 shows some example information tuples.

Table 6.1 Examples of BIM Information Tuples

Entity name	Entity Line ID	Attribute name list	Attribute value list
DOOR	6652	Globalid	1hOSvn6df7F8_7GcBWIRGQ
		Ownerhistory	Ownerhistory33
		Name	M_Single-Flush:1250mm x 2010mm:1250mm x 2010mm:146596
		Description	N/A
		Objecttype	1250mm x 2010mm
		Objectplacement	Localplacement6651
		Representation	Productdefinitionsshape6646
		Tag	146596
		Overallheight	2.0099999999999999
		Overallwidth	1.25
MEASUREWITHUNIT	20	Valuecomponent	0.01745329251994328
		Unitcomponent	Siunit18
SIUNIT	18	Dimensions	Dimensionalexponents39126
		Unittype	Planeangleunit
		Prefix	N/A
		Name	Radian

#### 6.2.1.2 Information Extraction

This phase aims to extract the entity name, entity line ID, attribute name list, and attribute value list of each entity in an .ifc file into the four-tuple format. The entities and attributes are extracted using their metadata (i.e., their EXPRESS data types). This allows full information extraction of all entities and their attributes using a small set of extraction rules. The Java Standard Data Access Interface (JSDAI) is also used for information extraction. This allows for information extraction based on any IFC schema (original or extended). The information extraction method includes five steps: (1) processing the IFC schema [in this dissertation, it is the extended IFC schema (Chapter 5)], (2) searching the IFC schema for the entities in

the .ifc file, (3) searching the IFC schema for the attributes of the entities in the .ifc file, (4) finding the data types of the attributes, and (5) extracting the values of the attributes based on their data types, using a set of extraction rules.

The IFC schema is processed into a collection of entity definitions. Each entity definition describes the name of an entity, the line ID of an entity, the data type of an entity, the attributes of an entity, and the supertypes of an entity. The collection of entity definitions supports the searching of entities and attributes in the following steps.

For searching the IFC schema for the entities in the .ifc file, each entity in an .ifc file is processed one by one. The definition of each entity is searched for in the collection of entity definitions. The information in its entity definition are used as follows: (1) the entity name and entity line ID are directly extracted into the first two elements of the entity's 4-tuple, respectively; (2) the data type of the entity (whether an aggregation data type or not) is used to decide on the use (or not) of recursive search and processing (on its subentities); and (3) the supertypes of the entity are used for extracting the information of the entity's attributes (described in the following step).

For searching the IFC schema for the attributes of the entities in the .ifc file, inheritance is taken into consideration. Searching the attributes of an entity includes searching both the direct attributes of the entity (i.e., explicit attributes), as well as the indirect attributes of the entity (i.e., the attributes of the entity's supertypes). The attributes of an entity's supertypes



are accessed in a recursive manner. For example, if an entity *B* is the supertype of an entity *A*, and an entity *C* is the supertype of the entity *B*; then at the time *A* is processed, in addition to the direct attributes of *A*, the attributes of *B* (supertype of *A*) and *C* (supertype of supertype of *A*) are accessed as well. The names of all direct and indirect attributes are extracted as a list in the third element of the entity's 4-tuple.

For finding the data types of the attributes of an entity, each attribute of the entity is searched for in the collection of entity definitions. The data type of each attribute is extracted from the corresponding entity definition and compared with the data types of the EXPRESS data definition language.

The values of the attributes are extracted based on their data types, using a set of extraction rules: (1) If an attribute is of a simple data type (i.e., integer, real, number, Boolean, logical, binary, or string), then the data value of the attribute is extracted. For example, if the "value component" attribute of a "measure with unit" (i.e., IFC entity to represent a quantity) is "0.6" (i.e., a real number), then "0.6" is extracted as a real number; (2) If an attribute is of the enumeration data type, then the enumeration value is extracted as a string. For example, if the "panel operation" attribute of a "door style" is "revolving" (i.e., an enumeration data), then "revolving" is extracted as a string; (3) If an attribute is of the entity data type, then the name and line ID (i.e., line number) of the entity are extracted and concatenated as a string. For example, if the "related space" attribute of a relation entity "covers spaces" is a "space" (i.e., an entity), then "space" (i.e., the entity name) and 1000 (i.e., the entity line ID) are extracted

and concatenated as a string “space1000;” and (4) If an attribute is of the aggregation data type (i.e., an aggregation of multiple values), then the multiple values in the aggregation are processed recursively according to their corresponding data types. For example, if the “related covering” attribute of a relation entity “covers spaces” is an aggregation (i.e., a set) of two “coverings” (i.e., entities), then the name and line ID of each covering entity are extracted and concatenated as a string (i.e., “covering2001,” “covering2002”).

The information extraction algorithm is illustrated in Figure 6.3. The dashed boxes show the two subroutines in the algorithm which allow recursive processing at the entity level and the attribute level, respectively. Table 6.1 shows some examples of the extracted information tuples.

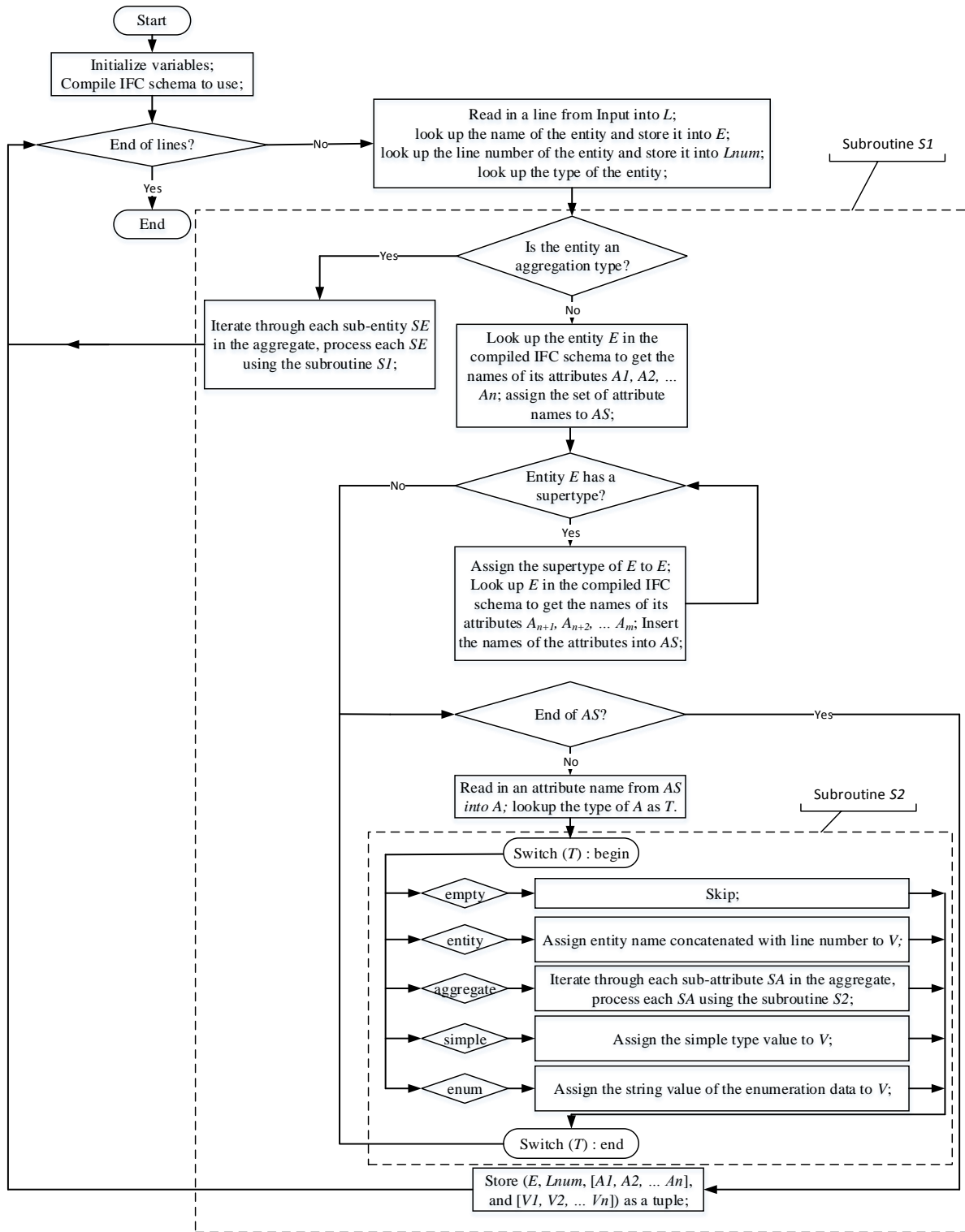


Figure 6.3 The Proposed BIM Information Extraction Algorithm

The Java Standard Data Access Interface (JSDAI) is used for information extraction.

Between the two data access types of early binding and late binding, late binding is selected, because it does not require one specific EXPRESS model (i.e., a specific IFC schema) but can be used for supporting information extraction based on any EXPRESS model (e.g., IFC 2X3, IFC\_2X3\_TC1, IFC4, any extended IFC). This use of late binding could support information extraction using the extraction rules based on EXPRESS data types, which is the key to avoid the need of knowing a specific BIM data schema beforehand. For all described five steps, JSDAI is used to access the entities and attributes in the .ifc file.

#### 6.2.1.3 Algorithm Implementation

The BIM information extraction algorithm was implemented using JSDAI (JSDAI 4.1.505.v201112201320) in Java Standard Edition Development Kit jdk1.7.0\_40 (Oracle 2015). The late binding data access type in JSDAI was used to access each entity and each attribute of an entity. Tail recursion was used for the recursive access of aggregation types of entities and attributes. For processing the input IFC schema (Step 1), the JSDAI Express compiler (JSDAI ExpressCompilerCore 4.1.11.v201112201318) was used to parse the input IFC schema and generate the collection of entity definitions in the form of Java classes and methods. For searching the IFC schema for the entities in the .ifc file (Step 2), the JSDAI model access methods are used to extract entity instances and JSDAI late binding entity access methods are used to access the entity definitions. The name of each entity was extracted by consulting the collection of entity definitions. The line number of each entity was extracted using the “getPersistentLabel” method of entity in JSDAI. For searching the

IFC schema for the attributes of the entities in the .ifc files (Step 3), finding the data types of the attributes (Step 4), and extracting the values of the attributes based on their data types (Step 5), JSDAI late binding attribute access methods are used to get the names, datatypes, and values of the attributes. The name of each attribute of an entity was extracted using the “getName” method of “explicit attribute” object in JSDAI. The data type of each attribute was extracted using the “getActualType” method in JSDAI. The value of each attribute was extracted using the “get” method of “entity definition” object in JSDAI.

#### 6.2.1.4 Evaluation Metrics

The evaluation is conducted by comparing the extracted BIM information (in the 4-tuple format) with those in a manually-developed gold standard. The gold standard includes all information tuples that represent all BIM information in a test case. Evaluation is conducted using the following two measures: precision and recall. Precision, here, is defined as the number of correctly extracted information tuples divided by the total number of information tuples extracted. Recall, here, is defined as the number of correctly extracted information tuples divided by the total number of information tuples that should have been extracted.

## **6.2.2 BIM Information Transformation**

### 6.2.2.1 Information Representation

This phase aims to define the ultimate representation format of the design information. The ultimate representation format is first order logic facts that could be directly used in a logic

programming platform for automated reasoning [i.e., logic facts that are aligned with the logic rules (e.g., for ACC reasoning it would be logic rules that represent regulatory requirements)]. Following the representation in Chapter 4, B-Prolog syntax is used. A logic fact is a concept fact or a relation fact. A concept fact defines a constant as an instance of a certain concept. For example, “window(window1)” defines the constant “window1” as an instance of the concept “window.” A relation fact defines a relationship between an instance of a concept and an instance of another concept or a value. For example, has(transverse\_reinforcement1, spacing1) defines the association relation between an instance of transverse reinforcement “transverse\_reinforcement1” and an instance of spacing “spacing1.” A logic rule defines an implication relation with one or more antecedents (i.e., predicates) that are conjoined and a single consequent (i.e., predicate). For example, the following logic rule (*LRI*) is a logic rule that defines the implication relation that “if spacing of transverse reinforcement is not greater than 8 inches, then the spacing is compliant with requirements” (Provision 1908.1.3 of IBC 2009):

```

“compliance_spacing_of_transverse_reinforcement(Spacing) :-
spacing(Spacing),transverse_reinforcement(Transverse_reinforcement),has(Transverse_reinf
orcement,Spacing),not greater_than(Spacing,quantity(8,inches)).”

```

#### 6.2.2.2 Information Transformation

This phase aims to transform the extracted tuple-represented entities and attributes into logic facts. A semantic rule-based method is used for the transformation. This information

transformation method includes two main steps: (1) initial transformation: transformation of entities and attributes into concept facts and relation facts; and (2) alignment transformation: further semantic transformation of the predicates of the concept facts and relations facts to be aligned with the predicates of the regulatory rules.

Initial transformation aims to transform the entities (concept or relation instances) and their attributes into concept facts and relation facts. Prior to transformation, entity names and attribute names are segmented in preparation for the following alignment transformation step.

For example, “TRANSVERSEREINFORCEMENT” is segmented to “transverse\_reinforcement” for the following alignment with *LRI*. Three main transformation rules are then used for the transformation: (1) an entity is transformed into a concept fact (i.e., a predicate) by using the name of the entity as the name of the predicate, and using the name of the entity concatenated with the line ID of the entity as the argument (i.e., an entity constant) of the predicate. The use of these entity line IDs satisfies three purposes: (a) identifying instances, (b) distinguishing instances, and (c) establishing links between the logic facts and their corresponding entities in their IFC source file. For example, in Figure 6.2, the pipe column entity is transformed into a concept fact “pipe\_column(pipe\_column40655),” with the name of the entity “pipe\_column” being the predicate name and the concatenation of the entity name and the entity line ID “pipe\_column40655” as the predicate argument; (2) an attribute of an entity is transformed into a relation fact (i.e., a predicate), using the name of the attribute preceded by “has\_” as the name of the predicate, using the corresponding entity

constant as the first argument of the predicate, and using the value of the attribute as the second argument of the predicate (if the value is not a reference to another entity). For example, in Table 6.1, the attribute name “overallheight” (in the attribute name list) for the “door” entity is transformed into a relation fact “has\_overall\_height(door6652, 2.0099999999999999);” and (3) if the value of an attribute is a reference to another entity, then the referred entity constant is used as the second argument of the predicate. For example, in Table 6.1, the attribute name “ownerhistory” (in the attribute name list) for the “door” entity is transformed into a relation fact with the referred entity constant owner\_history33 as the second argument: “has\_owner\_history(door6652,owner\_history33).”

Alignment transformation aims to further transform the predicates of the logic facts (concept facts and relations facts) to be aligned with the predicates of the logic rules. For example, the predicates of the logic facts *F1* to *F4* need to be aligned with the predicates of the logic rule *LR2*. A set of semantic transformation (SeTr) rules are used to conduct the transformation. Two types of SeTr rules (in the format of logic rules) are used: static SeTr rules and dynamic SeTr rules. A static rule is defined as a rule that only uses static predicates. A dynamic rule is defined as a rule that uses at least one dynamic predicate(s). A static predicate is a predicate that cannot be updated during execution. A dynamic predicate is a predicate that can be updated during execution. Static rules are used when data (constants) are transformed from an argument in a predicate to an argument in another predicate. For example, the SeTr rule *R1* transforms logic facts *F1* to *F4* into one single logic fact *F5*. Dynamic rules are used when



data (constants) are transformed from an argument in a predicate to the name of another predicate. For example, the SeTr rule *R2* further transforms the logic fact *F5* to logic fact *F6*. As a result of the transformation, logic fact *F6* becomes aligned with logic rule *LR2*. The use of SeTr rules enables the use of domain knowledge for the flow of design information. For example, based on the domain knowledge that “skylight” is a subconcept of “window,” a static SeTr rule could be formalized as “window(*X*) :- skylight(*X*),” which enables the instances of skylight to be able to instantiate rules for windows. In the state-of-the-art ACC efforts that use ontology as the main representation of information, such function is achieved through matching ontologies [e.g., through similarity value measurements (Yurchyshyna et al. 2007)].

- SeTr *R1*: *acc\_bi\_relation*(*Name*, *Y*, *Z*) :- *acc\_bi\_relation*(*X*, *has\_type\_name*(*X*,*Name*), *has\_relating\_element*(*X*,*Y*), *has\_related\_element*(*X*,*Z*).
- SeTr *R2*: *SeTrRule2* :- *findall*((*Term*, *Name*, *Y*, *Z*), *acc\_bi\_relation*(*Name*, *Y*,*Z*), *Xs*), *sort*(*Xs*, *Xs1*), *foreach*((*Term*, *Name*,*Y*,*Z*) in *Xs1*, (*Term* =.. [*Name*, *Y*, *Z*], *assert*(*Term*))).
- *F1*: *acc\_bi\_relation*(*acc\_bi\_relation40657*).
- *F2*: *has\_type\_name*(*acc\_bi\_relation40657*,*used\_in*).
- *F3*: *has\_relating\_element*(*acc\_bi\_relation40657*,*pipe\_column40655*).
- *F4*: *has\_related\_element*(*acc\_bi\_relation40657*,*basement40656*).
- *F5*: *acc\_bi\_relation*(*used\_in*, *pipe\_column40655*, *basement40656*).

- *F6: used\_in(pipe\_columns40655, basement40656).*
- *LR2: compliance\_minimum\_diameter\_of\_pipe\_column(Minimum\_diameter):-  
pipe\_column(Pipe\_column),used\_in(Pipe\_column,Basement),basement(Basement),as(  
Pipe\_column,Secondary\_steel\_member),secondary\_steel\_member(Secondary\_steel\_  
member),has(Pipe\_column,Minimum\_diameter),minimum\_diameter(Minimum\_diame  
ter),greater\_than\_or\_equal(Minimum\_diameter,quantity(4,inches)).*

### 6.2.2.3 Algorithm Implementation

The information transformation algorithm was implemented using the Java Standard Edition Development Kit jdk1.7.0\_40 (Oracle 2015). The segmentation of entity names and attribute names was implemented using regular expression-based matching methods in Java. The segmented entity names and attribute names were stored in a parallel list. The transformation of tuple-represented entities and attributes into logic facts was implemented using string processing methods in Java. The SeTr rules were implemented in B-Prolog logic programming language (Zhou 2012).

### 6.2.2.4 Evaluation Metrics

The evaluation is conducted by comparing the transformed BIM information (represented as logic facts, including concept facts and relation facts) with those in a manually-developed gold standard. The gold standard includes the ground truth of concept facts and relation facts. For practicality, a Python program (Python v.2.7.3) was developed to conduct this comparison (matching) in an automated way. The evaluation was conducted using the

following measures: precision and recall. Precision, here, is defined as the number of correctly transformed logic facts divided by the total number of logic facts transformed. Recall, here, is defined as the number of correctly transformed logic facts divided by the total number of logic facts that should have been transformed.

### **6.3 Experimental Testing and Evaluation**

To evaluate the proposed BIM information extraction and information transformation method, a BIM test case was used to test: (1) the extraction of the BIM information from the .ifc file into the information tuples; and (2) the transformation of the information tuples into logic facts. An extended IFC schema was used for the information extraction. The extracted information tuples and transformed logic facts were compared with those in a manually-developed gold standard and were evaluated in terms of precision and recall (as described in the Sections 6.2.1.4 and 6.2.2.4). In developing the gold standard of logic facts, the logic facts were aligned with a regulatory rule testing set (that represent regulatory requirements, which are intended to support automated compliance reasoning). The test case design information were at a level of development (LOD) of 400 (BIMForum 2013). For example, in addition to height and thickness of walls (i.e., LOD 300), reinforcement information of walls were included too (Figure 6.4).

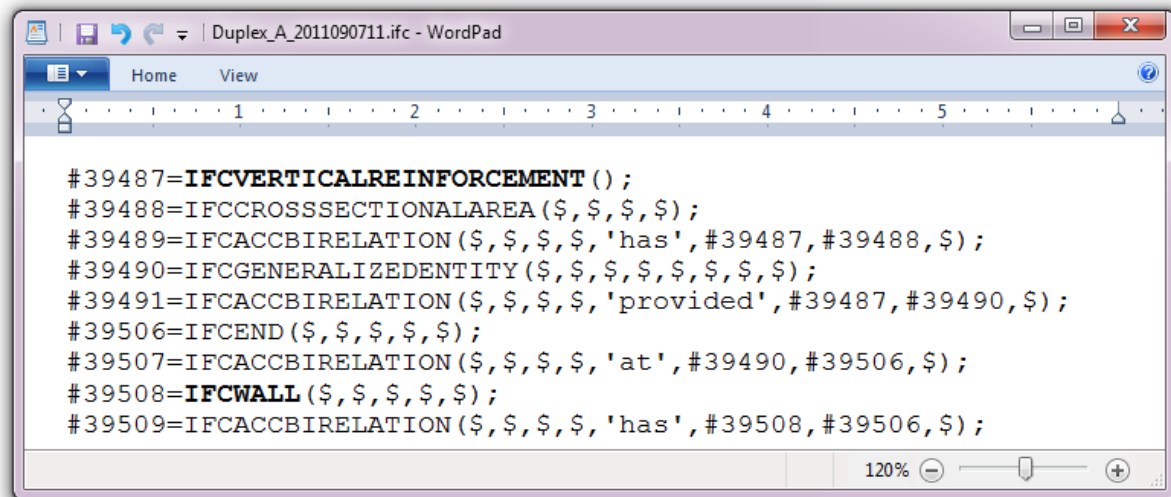


Figure 6.4 Example Reinforcement Information of Wall in the BIM Test Case

### 6.3.1 BIM Test Case

A BIM test case based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences (East 2013) was developed. Design information were added in the BIM model, based on the extended IFC schema in Section 5.3.4. The test case included design information for each provision in Chapter 19 of IBC 2009. The design information included both compliant and noncompliant design information to test the performance of information extraction and transformation in both scenarios. For example, the following regulatory provision (*RPI*) is a complex provision that contains three quantitative requirements: *“In dwellings assigned to Seismic Design Category D or E, the height of the wall shall not exceed 8 feet (2438 mm), the thickness shall not be less than 7 1/2 inches (190 mm), and the wall shall retain no more than 4 feet (1219 mm) of unbalanced fill”* (Provision 1908.1.8 of IBC 2009). Thus, five information sets were created for *RPI* which correspond to

the scenarios that (1) only height is noncompliant, (2) only thickness is noncompliant, (3) only unbalanced fill is noncompliant, (4) all three attributes are noncompliant, and (5) no attributes are noncompliant. In total, 146 design information sets were created for the 63 provisions in Chapter 19 of IBC 2009. While any tool that supports accessing IFC information can fulfill the purpose, JSDAI was selected to add design information to the .ifc file of the test case. Figure 6.5 provides a snapshot of the software interface showing the addition of the following example concept and relation facts: an instance of “pipe column,” an instance of “basement,” an instance of “minimum diameter,” the “used\_in” relation between the pipe column instance and the basement instance, the “has” relation between the “pipe column” instance and the “minimum diameter” instance, a quantity with the value of “3.0” and the unit of “inches,” and the quantitative relation between the “minimum diameter” instance and the quantity.

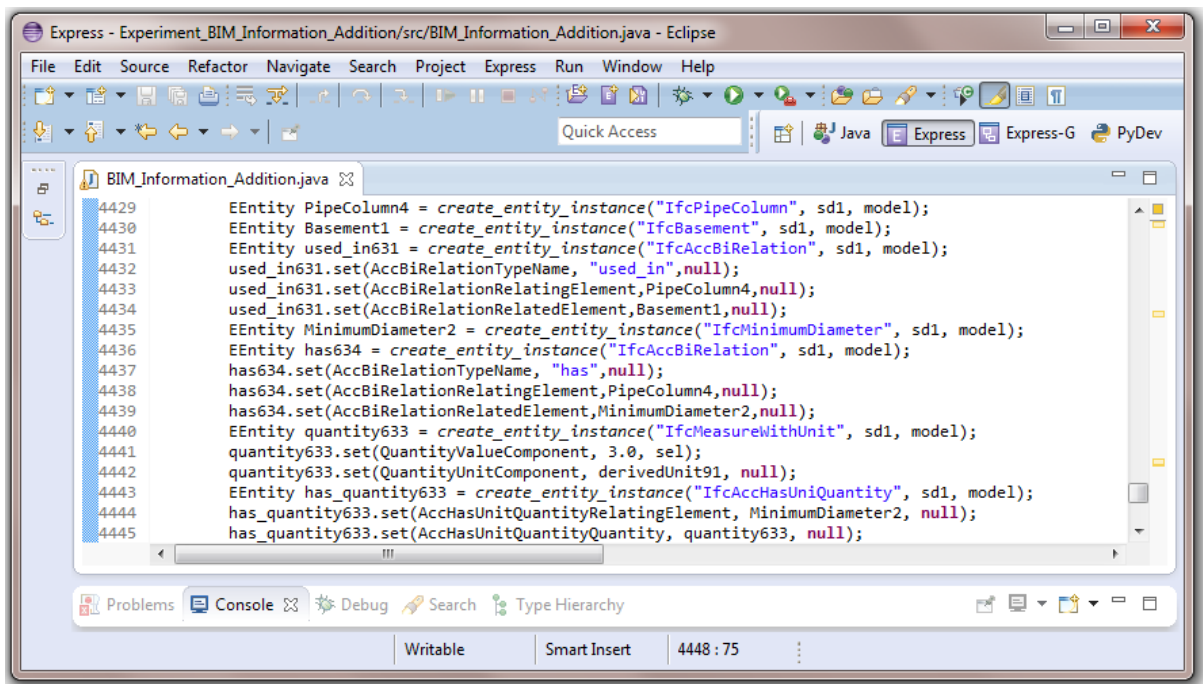


Figure 6.5. Software Interface Snapshot Showing an Example of the Addition of Concept and Relation Facts for the BIM Test Case Development

### 6.3.2 Results and Discussion

The information extraction results are summarized in Table 6.2. A total of 1,603 information tuples were extracted and compared to those in the gold standard. A 100% (95% confidence interval [99.8%, 100%]) precision and recall was achieved.

Table 6.2 BIM Information Extraction Testing Results

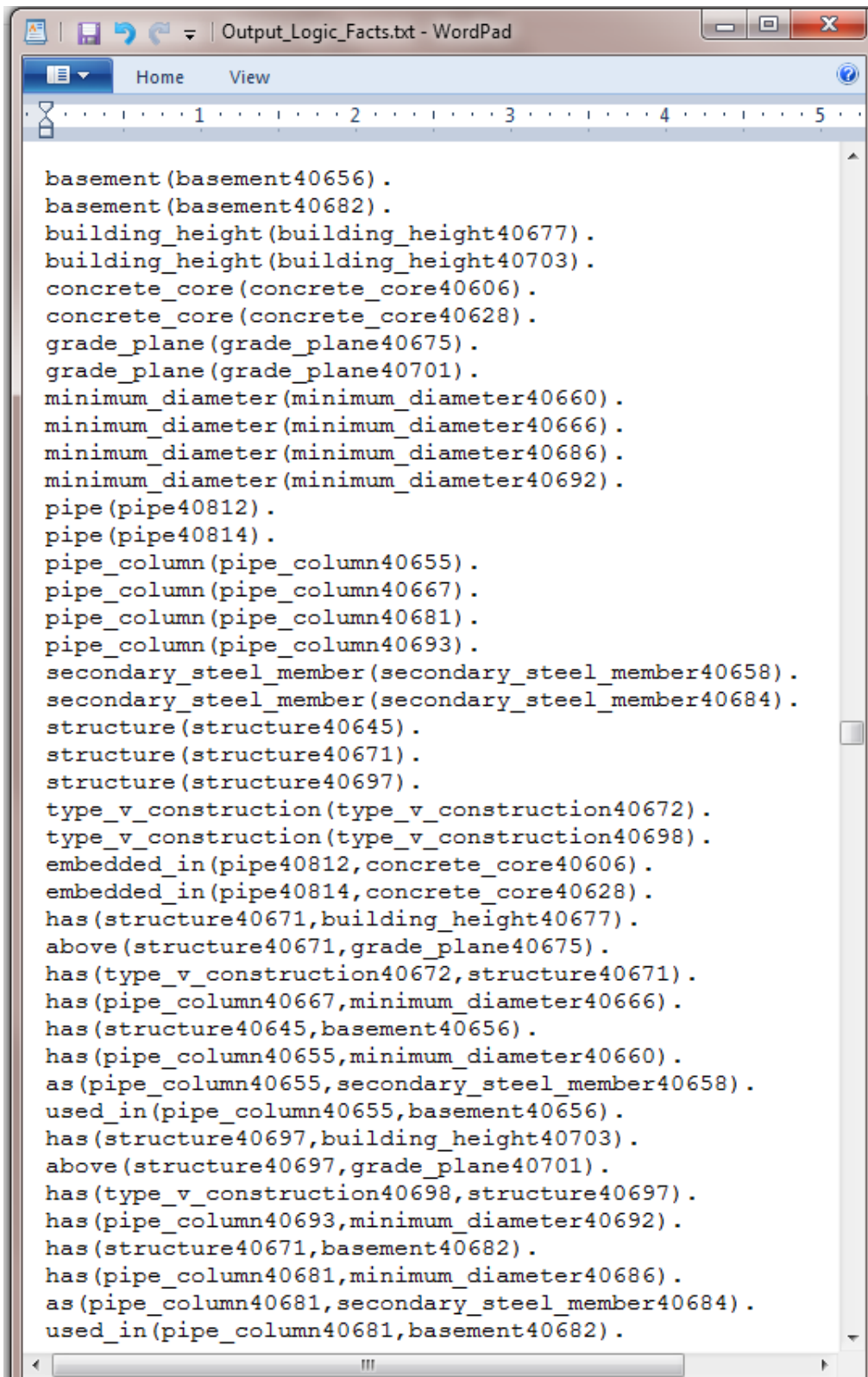
Extracted item	Number correctly extracted	Total number extracted	Total number in gold standard	Precision	Recall
Information tuple	1,603	1,603	1,603	100%	100%

The information transformation results are summarized in Table 6.3. A total of 4,075 and 1,496 logic facts were transformed before and after the alignment transformation, respectively, and were compared to those in the gold standard. A 100% (95% confidence

interval [99.8%, 100%]) precision and recall was achieved. Figure 6.6 provides a snapshot of the output of the developed BIM information extraction and transformation software, showing partial transformation results (i.e., logic facts) after alignment transformation. These logic facts are ready for importing in any logic programming platform for automated logic-based reasoning.

Table 6.3 BIM Information Transformation Testing Results

Transformed item	Number transformed before alignment transformation	Number correctly transformed after alignment transformation	Total number transformed after alignment transformation	Total number in gold standard	Precision	Recall
Concept facts	810	688	688	688	100%	100%
Relation facts	3,265	808	808	808	100%	100%
Logic facts (total)	4,075	1,496	1,496	1,496	100%	100%



```
basement(basement40656).
basement(basement40682).
building_height(building_height40677).
building_height(building_height40703).
concrete_core(concrete_core40606).
concrete_core(concrete_core40628).
grade_plane(grade_plane40675).
grade_plane(grade_plane40701).
minimum_diameter(minimum_diameter40660).
minimum_diameter(minimum_diameter40666).
minimum_diameter(minimum_diameter40686).
minimum_diameter(minimum_diameter40692).
pipe(pipe40812).
pipe(pipe40814).
pipe_column(pipe_column40655).
pipe_column(pipe_column40667).
pipe_column(pipe_column40681).
pipe_column(pipe_column40693).
secondary_steel_member(secondary_steel_member40658).
secondary_steel_member(secondary_steel_member40684).
structure(structure40645).
structure(structure40671).
structure(structure40697).
type_v_construction(type_v_construction40672).
type_v_construction(type_v_construction40698).
embedded_in(pipe40812,concrete_core40606).
embedded_in(pipe40814,concrete_core40628).
has(structure40671,building_height40677).
above(structure40671,grade_plane40675).
has(type_v_construction40672,structure40671).
has(pipe_column40667,minimum_diameter40666).
has(structure40645,basement40656).
has(pipe_column40655,minimum_diameter40660).
as(pipe_column40655,secondary_steel_member40658).
used_in(pipe_column40655,basement40656).
has(structure40697,building_height40703).
above(structure40697,grade_plane40701).
has(type_v_construction40698,structure40697).
has(pipe_column40693,minimum_diameter40692).
has(structure40671,basement40682).
has(pipe_column40681,minimum_diameter40686).
as(pipe_column40681,secondary_steel_member40684).
used_in(pipe_column40681,basement40682).
```

Figure 6.6 Software Output Snapshot Showing Partial BIM Information Transformation Results (Logic Facts) After Alignment Transformation



## **7 CHAPTER 7 – LOGIC-BASED INFORMATION REPRESENTATION AND COMPLIANCE REASONING SCHEMA**

### **7.1 Comparison to the State of the Art**

The state-of-the-art ACC in the AEC industry mostly relies on the use of proprietary rules for representing regulatory requirements. For example, the CORENET project coded regulatory rules in C++ programs, the Solibri model checker uses a proprietary proforma-based format to code regulatory rules, and several ACC research efforts coded regulatory rules for specific subdomains such as fall protection (Zhang et al. 2013), building envelope performance (Tan et al. 2010), and accessibility (Lau and Law 2004). Such rules could be very effective in reasoning about compliance with a specific set of requirements and specific regulatory sections in a certain period of time, but such rigid and static representation requires great effort in (1) adaptation to different regulatory codes/sections and (2) maintenance/update across different time periods and in response to code revisions/updates. The use of proprietary rules, thus, becomes effort-intensive and time-consuming because of the large number of codes/regulations and the frequent revisions/updates of codes/regulations (Delis and Delis 1995; Dimyadi and Amor 2013).

To avoid the reliance on proprietary rules, few researchers explored the development of generalized representations/schemas for the formalization of regulatory requirements. For example, Hjelseth and Nisbet (2011) proposed the requirement, applies, select, and exception

(RASE) method to capture and represent regulatory requirements in the AEC industry; Yurchyshyna et al. (2010; 2008) developed a conformity-checking ontology that captures regulatory information together with building-related knowledge and expert knowledge on checking procedures; Beach et al. (2013) extended the RASE method for representing requirements in the UK's building research establishment environmental assessment method (BREEAM) and the code for sustainable homes (CSH); and Dimyadi et al. (2014) utilized the drools rule language (DRL) to represent regulatory rules.

These efforts contributed to the improvement of flexibility and reusability of regulatory representations for ACC. However, they are still limited in terms of: (1) automated regulatory information extraction and transformation: the state of the art in ACC still requires major manual efforts in extracting regulatory information from textual regulatory documents and transforming/encoding these information into a computer-processable rule format; and (2) automated reasoning: the state-of-the-art ACC efforts still use ad-hoc reasoning schema/methods, with lack of support for complete automation in reasoning. For example, in Hjelseth and Nisbet (2011), (1) the extraction of regulatory information and their encoding into the RASE representation is still manually conducted and (2) no specific mechanism for reasoning about the RASE-represented regulatory requirements was proposed. For the ontology-based effort by Yurchyshyna et al. (2010; 2008), (1) the extraction of regulatory information and their encoding into SPARQL Protocol and RDF Query Language (SPARQL) queries is also manually conducted and (2) the reasoning in their ontology-centered approach

was implemented by matching resource description framework (RDF)-represented design information with SPARQL queries-represented regulatory information, but a set of expert rules need to be manually defined through document annotations (i.e., annotations by content and external sources) to organize the SPARQL queries and enable reasoning, resulting in ad-hoc reasoning and lack of full automation. In the work by Beach et al. (2013) and Dimyadi et al. (2014), (1) the extraction of regulatory information and their encoding into the extended RASE representation and DRL rules, respectively, is still manually conducted by experts, and (2) the mechanism of reasoning (e.g., sequence of rule execution) was not specified.

There is, thus, a need for a “standard, generalized approach for formally representing building regulations in a digital format that would facilitate a variety of forms of reasoning about those codes in combination with digital building information models” (Garrett and Palmer 2014). The needed representation approach should also facilitate automated information extraction and information transformation to support complete automation of ACC.

In an automated reasoning system, the representation schema and reasoning mechanism influence each other. Reasoning needs affect the requirements and structure of the representation and successful reasoning depends on appropriate representations. Finding the right representation is, thus, a key to successful reasoning (Bundy 2013).

FOL representation and reasoning can provide a generalized reasoning method to facilitate complete automation in ACC reasoning (Kerrigan and Law 2003; Halpern and Weissman

2007). A limited number of research efforts have used FOL-based representation and reasoning in the AEC industry. Jain et al. (1989) introduced an information representation method that used FOL-based reasoning to support structural design. Rasdorf and Lakmazaheri (1990) used a FOL approach to (1) designing structural members according to the American Institute of Steel Construction (AISC) specifications, and (2) checking the compliance of designed structural members with the specifications. Kerrigan and Law (2003) used a FOL approach to supporting regulatory compliance assessment with Environmental Protection Agency (EPA) regulations. Outside of the AEC industry, a number of efforts have proposed the use of FOL for supporting conformance reasoning, such as compliance checking (Awad et al. 2009), policy auditing (Garg et al. 2011), and law verification (DeYoung et al. 2010). Despite the importance of these efforts, there are three main knowledge gaps in the area of FOL-based ACC. First, there is a lack of knowledge on which assumption is better-suited for ACC – a closed world assumption or an open world assumption in noncompliance detection. For example, Rasdorf and Lakmazaheri (1990) followed a closed world assumption for noncompliance detection, while Kerrigan and Law (2003) used an open world assumption; but there are no efforts that compared both assumptions in terms of performance in ACC applications. Second, there is a lack of knowledge on how to use a closed world assumption model in noncompliance detection without introducing many false positives. A closed world assumption can typically lead to a high number of false positives, because missing information would result in failure to deduce

compliance. For example, Denecker et al. (2011) chose to drop the closed world assumption because they could not avoid the false positives caused by missing information. Third, there is a need for further ACC-specific computational and reasoning support for using existing logic-based reasoners. For instance, there is a need for further built-in logic rules or functions to identify the sequence of checking different regulatory requirements. For example, Kerrigan and Law (2003) used control elements (i.e., functions) to specify the sequence of checking provisions for each regulation; but, this approach is limited because these control elements must be specified by a domain expert for every regulation.

## **7.2 Proposed Information Representation and Compliance Reasoning Schema**

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

ontology-based LCs and the activation conditions were used in Alternative I to avoid the problem of missing information causing false positives in closed world assumption schemas. A support module was also developed, as part of the schema, to provide ACC-specific reasoning support.

As such, the proposed IRep and CRes schema is composed of two main modules (as per Figure 7.1): a data module (which is dynamic) and a support module (which is fixed). The data module consists of information LCs. An information LC could be a regulatory information LC or a design information LC. Regulatory information LCs and design information LCs are used to represent applicable regulatory requirements and existing design information, respectively. They are automatically created/updated from semantic information elements that are automatically extracted from corresponding regulatory documents (e.g., IBC 2009) and design information sources (e.g., a BIM), and automatically transformed into the LC format. Information LCs are, thus, dynamically-created/updated based on applicable regulatory documents (i.e., building codes) and design information sources.

The support module was developed to provide reasoning support to the data module, and consists of functional built-in LCs. The functional built-in LCs are used for implementing basic arithmetic functions (such as unit conversion) and defining reasoning sequences/strategies (such as the sequence of checking different regulatory requirements). The functional built-in LCs would be predefined (built-in) in an ACC system and, thus, would be fixed across different compliance checking instances.

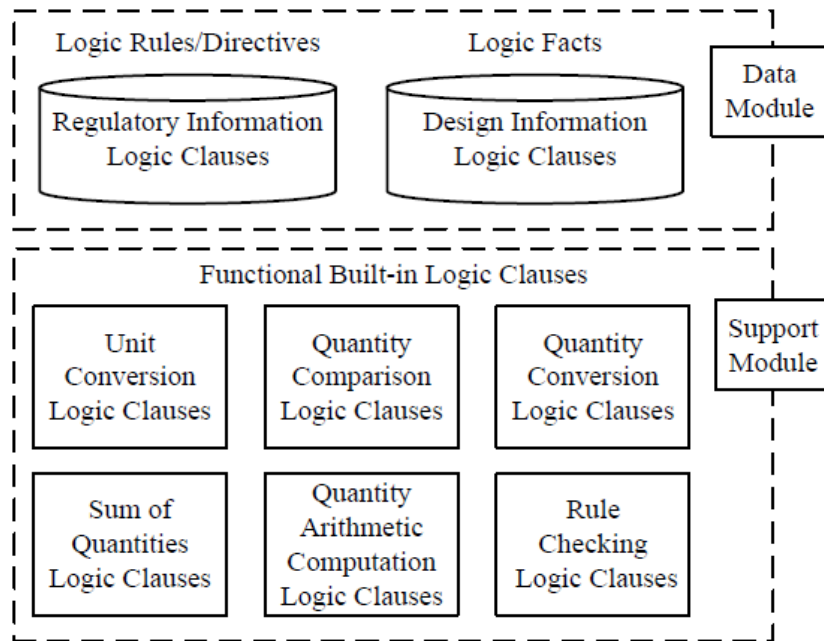


Figure 7.1 The Proposed IRep and CRes Schema

### 7.2.1 Main Features of the Proposed Information Representation and Compliance Reasoning Schema

The proposed IRep and CRes schema is characterized by three main features. First, the representation is semantic. A semantic representation is essential to leverage domain knowledge in the reasoning process in order to handle the complex relations involved in compliance reasoning and enable deep reasoning. This is important because the relations in regulatory provisions could be very complex. For example, Figure 7.2 shows the many relations involved in one single regulatory provision in IBC 2006, leading to a very complex regulatory provision. The semantic representation is supported by an ontology that is used in a deep manner (i.e., the ontology supports deep information extraction, information

transformation, and the IRep and CRes schema). The semantic representation also facilitates human understandability and interpretability of the formal representation, which is essential to facilitate usability and allow for human testing and verification of the information representation and the reasoning results. Second, the representation is logic-based. A logic-based representation was selected to take advantage of the well-matured logic-based reasoning techniques. Logic-based reasoning is well-suited for ACC problems because: (1) The binary nature (“satisfy or fail to satisfy”) of the smallest reasoning units (i.e., LCs) fits the binary nature (“compliance or noncompliance”) of ACC tasks; (2) A variety of automated reasoning techniques such as search strategies and unification mechanisms are available in ready-to-use reasoners; (3) Many formally-defined logics have sufficient expressiveness to represent concepts and relations involved in ACC; and (4) once the information is properly represented in a logic format, the reasoning becomes completely automated. Among the existing types of logic, FOL was selected because “a FOL sentence can mostly be translated into an English sentence which is guaranteed to be true if and only if the FOL sentence is true in interpretation” (Hodges 2001). This: (1) enables isomorphism: one-to-one mapping between an English regulatory requirement and a logic clause (LC); and (2) as a result allows for traceability: maintaining traceability is important to identify the sources of LCs and, thus, to facilitate human verification and ensure trustworthiness of the LCs and the results. Third, the representation is generalized and flexible. The generalization and flexibility are achieved through generalized regulatory compliance checking concepts and flexible semantic



information elements. Generalized regulatory compliance checking concepts (e.g., “subject” and “compliance checking attribute”) are used, which enables the schema to represent regulatory provisions of any type/topic (e.g., building envelope performance, facility accessibility). Flexible information elements (e.g., “subject restriction,” as represented in Chapter 3 and Chapter 4) are used, which enables the schema to represent all information (i.e., all concepts and relations) in a regulatory provision regardless of the length and complexity of the provision (sentence). Generalization and flexibility are important to sustain utility and robustness of the proposed schema across different types of regulatory documents and different types of provisions.

**Regulatory Provision from IBC 2006**

The minimum required net free ventilating area shall be 1/300 of the area of the space ventilated, provided a vapor retarder having a transmission rate not exceeding 1 perm in accordance with ASTM E 96 is installed on the warm side of the attic insulation and provided 50 percent of the required ventilating area provided by ventilators located in the upper portion of the space to be ventilated at least 3 feet above eave or cornice vents, with the balance of the required ventilation provided by

**Semantic Representation of the Regulatory Provision**

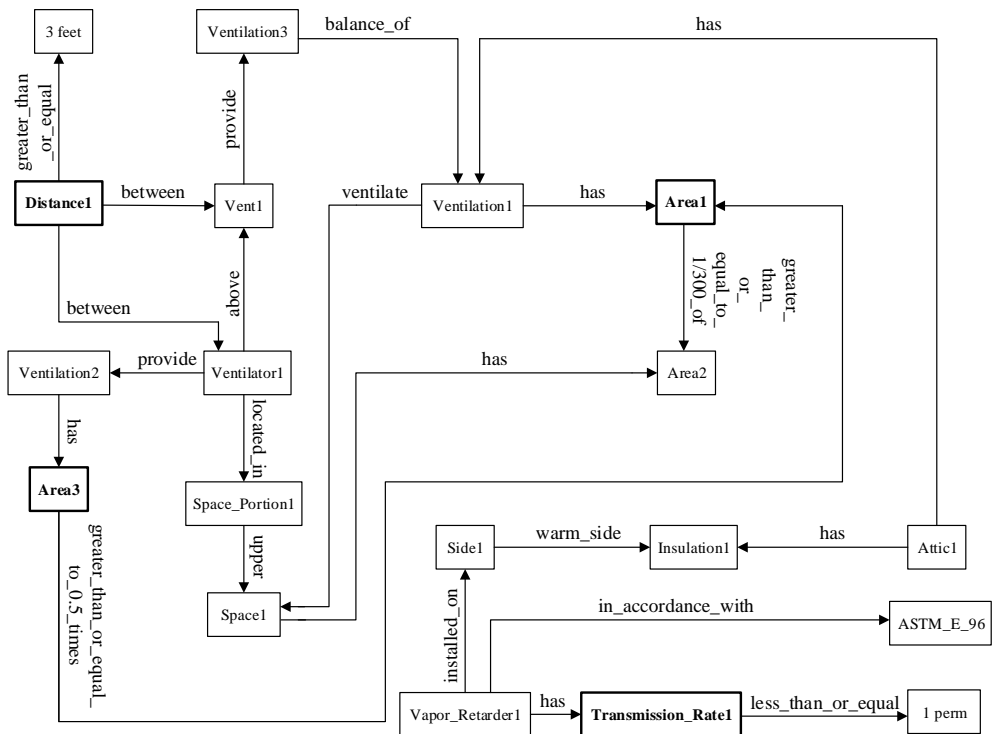


Figure 7.2 An Example Provision and the Involved Relations

## 7.2.2 Semantic Information Elements and their Link to the Logic Clauses

The predicates in the LCs are semantic; they are linked to a set of “semantic information elements” (presented in Chapter 3). The semantic information elements are, in turn, linked to a building ontology. A semantic information element (see Figure 7.3) is a “subject,” “compliance checking attribute,” “deontic operator indicator,” “quantitative relation,” “comparative relation,” “quantity value,” “quantity unit,” “quantity reference,” “restriction,” or “exception” (see Chapter 3 for definitions). A semantic representation is essential to (1) distinguish the ACC-specific meaning of the different predicates by linking the predicates to the semantic information elements, and (2) associate further AEC-specific meaning to the different predicates by linking the semantic information elements to the ontology concepts and relations. For example, by linking the predicate “transverse\_reinforcement(transverse\_reinforcement)” to the “subject” and “spacing(spacing)” to the “compliance checking attribute,” we can distinguish that the former is the subject of the regulatory requirement, while the latter is the compliance checking attribute of this subject. In turn, by linking the “transverse\_reinforcement” (i.e., name of the predicate) to ontology concepts, we can further recognize that “transverse\_reinforcement(transverse\_reinforcement)” is a type of “building element.” The use of semantic-based LCs also plays a central role in identifying and formalizing the activation conditions (as described in the following section).

The recognition, extraction, and transformation of the semantic information elements into the

predicates are automatically conducted during preceding information extraction and information transformation processes – both aided by the ontology for capturing the semantic features of the text. The details of the information extraction and information transformation methods and algorithms are presented in Chapter 3 and Chapter 4, respectively.

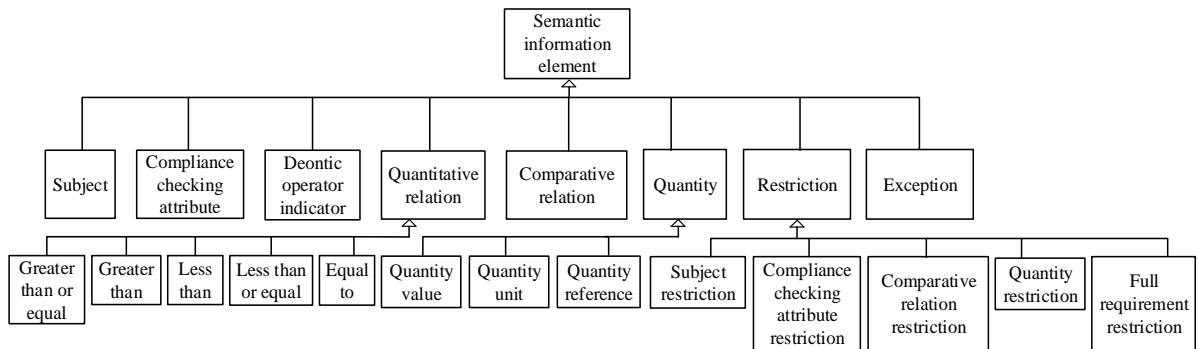


Figure 7.3 Semantic Information Element

## 7.2.3 Information Logic Clauses

### 7.2.3.1 Regulatory Information Logic Clauses

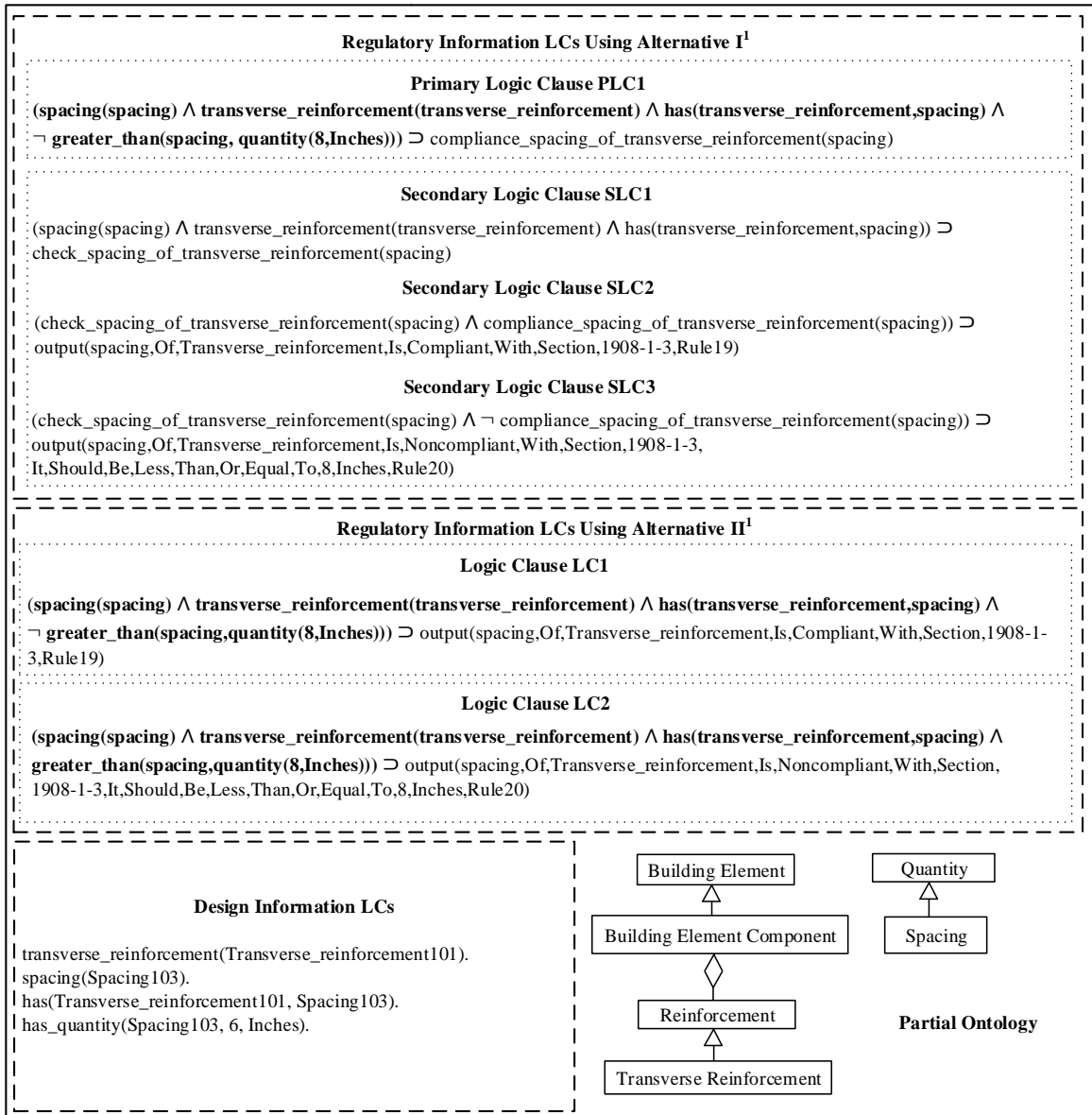
Two alternative subschemas were developed. Alternative I implements a closed world assumption (i.e., the assumption that what is not known to be true is false) for noncompliance detection, which means that the design information that are not found to be compliant are regarded as noncompliant. Alternative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

are executed.

### Alternative I

In Alternative I, regulatory information LCs are represented using logic rules. Two types of regulatory information LCs are represented (as per Figure 7.4): primary regulatory information LCs and secondary regulatory information LCs (are called primary and secondary LCs hereafter). Each regulatory requirement (a provision could include multiple requirements as explained above) is represented as one primary LC and is supported by three secondary LCs. For example (see Figure 7.4), regulatory provision *RP2* (here the provision has one requirement about “spacing”) is represented using PLC1, SLC1, SLC2, and SLC3.

- *RP2: “Spacing of transverse reinforcement shall not exceed 8 inches” (from Provision 1908.1.3 of IBC 2009).*



<sup>1</sup>All variables are universally quantified, but quantifiers are not shown.

Figure 7.4 Alternative I and Alternative II of the Proposed IRep and CRes Schema (in First Order Logic)

A primary LC is the core representation of a requirement. It represents the compliance case. The premise of a primary LC represents the conditions of the requirement (e.g., the conditions that would make the spacing of transverse reinforcement compliant) and the

conclusion of a primary LC represents the consequent result which is the compliance with the requirement (e.g., the compliance of the spacing of the transverse reinforcement). As such, compliance is deduced from primary LCs (compliance case), while noncompliance cases are inferred based on compliance cases (i.e., if a subject is not compliant with a primary LC, then it is noncompliant – following a closed-world assumption). As mentioned in the preceding subsection, the predicates in the primary LCs are linked to “semantic information elements,” where the instances of these semantic information elements were automatically recognized, extracted, and transformed into these LCs during the preceding ontology-based information extraction and information transformation processes. Semantic information elements are, in turn, linked to ontology concepts and relations. For example (see Figure 7.4), the predicates to the left of “ $\supset$ ” in the primary rule PLC1 are the premise conditions of the LC, where each predicate represents an ontology concept or an ontology relation (a partial view of the ontology is also shown in Figure 7.4). For example, the predicate “transverse\_reinforcement(transverse\_reinforcement)” represents the concept “transverse reinforcement” (subconcept of “building element” which is a “subject”), the predicate “spacing(spacing)” represents the concept “spacing” (subconcept of “quantity,” which is a “compliance checking attribute”), and the predicate “has(transverse\_reinforcement, spacing)” represents the relation “transverse reinforcement”-“has”-“spacing,” which is a relation between a “subject” and a “compliance checking attribute.” The conclusion of a primary LC is one single predicate that takes the following standardized pattern:

“compliance\_*ComplianceCheckingAttribute\_of\_Subject(complianceCheckingAttribute)*,”

where the *ComplianceCheckingAttribute* and the *Subject* are the “compliance checking attribute” and the “subject” of the requirement, respectively. For example (see Figure 7.4), the following predicate represents the conclusion of PLC1, which is automatically constructed during information transformation from the extracted “subject” (“transverse reinforcement”) and the extracted “compliance checking attribute” (“spacing”) of the requirement: “compliance\_spacing\_of\_transverse\_reinforcement(spacing).” If multiple regulatory requirements exist in one regulatory provision, each of the regulatory requirements is represented in a separate primary LC and reported separately. For example, for regulatory provision *RP1*, the “height,” “thickness,” and “unbalanced\_fill” of the “wall” instance are represented in three separate primary LCs and reported separately.

- *RP1: “In dwellings assigned to Seismic Design Category D or E, the height of the wall shall not exceed 8 feet (2438 mm), the thickness shall not be less than 7 1/2 inches (190 mm), and the wall shall retain no more than 4 feet (1219 mm) of unbalanced fill.” (Provision 1908.1.8 of IBC 2009)*
- *RP2: “Spacing of transverse reinforcement shall not exceed 8 inches.” (Provision 1908.1.3 of IBC 2009)*

A regulatory document includes one or more regulatory provisions (e.g., a sentence in IBC 2009), and a regulatory provision includes one or more regulatory requirements [(e.g., a sentence describing minimum requirements of both width and height of a door includes two

requirements (one requirement about the width and one about the height)]. For example, regulatory provision *RP1* includes three regulatory requirements about the “height,” “thickness,” and “unbalanced fill” of the “wall,” while *RP2* includes only one regulatory requirement about the “spacing” of the “transverse reinforcement.”

Each primary LC is supported by three secondary LCs: (1) one for representing the conditions that activate the checking of the requirement, and (2) two for representing the consequences of the compliance checking result. Activation conditions (1) help prevent missing information from leading to false positives because missing information would lead to failure in activation, and (2) avoid exhaustive search over all design information LCs and thus lead to higher computational efficiency (during software implementation). The activation conditions for each regulatory requirement define the premise conditions of the requirement, which are generated from the respective primary LC by separating the premise conditions [e.g., “spacing(spacing)  $\wedge$  transverse\_reinforcement(transverse\_reinforcement)  $\wedge$  has(transverse\_reinforcement, spacing)”] from the consequent prescription [e.g., “ $\neg$ greater\_than(spacing, quantity(8,Inches))”]. The semantic representation helps recognize the premise conditions of a regulatory requirement in a primary LC through the semantic information elements. The consequences for each requirement are also linked to instances of semantic information elements that are automatically recognized, extracted, and transformed into these secondary LCs during information extraction and information transformation processes. A “compliance checking result” could be compliance or noncompliance, and a



“compliance checking consequence” is the outcome or effect of the “compliance checking result” such as a suggested corrective action. For example, the checking of the regulatory requirement represented in PLC1 is activated using SLC1. If any information in the body of SLC1 is missing (e.g., the relation between the spacing and the transverse reinforcement is missing), then the checking with PLC1 would not be activated, which would avoid a blind activation of SLC3 that would lead to a false positive noncompliance. For the checking result, using SLC2 and SLC3, an output message including whether the result is compliant or noncompliant is printed out, together with the relevant provision number (i.e., “1908.1.3”) and the regulatory requirement rule ID. If the result is noncompliant, a corrective suggestion on how to fix the noncompliance is provided (i.e., “the spacing should be less than or equal to 8 inches”). The modeling of compliance checking consequences allows for deep compliance reasoning (i.e., not only finding instances of noncompliance but also offering an analysis of the noncompliance and providing suggestions for corrective actions).

## Alternative II

In Alternative II, each regulatory requirement is represented using two logic rules, one for representing the compliance case and one for explicitly representing the noncompliance case. As such, noncompliance cases are explicitly represented instead of being inferred based on compliance cases – following an open world assumption). For example, in Figure 7.4, LC1 and LC2 are two LCs representing the compliance case and noncompliance case of a regulatory requirement, respectively. As such, the premise of LC1 represents the conditions

of compliance with a requirement, whereas that of LC2 represents the conditions of noncompliance with the same requirement. Different from Alternative I, there is no need to use secondary LCs for representing activation conditions and consequences of compliance checking results, because compliance and noncompliance cases are represented separately. As such, the conclusions of LC1 and LC2, represent both the “compliance checking results” (compliant or noncompliant) and the “compliance checking consequences” (e.g., a corrective suggestion on how to fix the noncompliance). Similar to Alternative I, predicates in the LCs link to ontology concepts or relations.

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

### 7.2.3.2 Design Information Logic Clauses

Design information LCs, in both Alternative I and Alternative II, are represented using logic facts. Each single design fact (e.g., Transverse\_reinforcement101 is an instance of transverse reinforcement) is represented as one single design information LC (logic fact). A design fact could be a concept fact or a relation fact. A concept fact is represented by a design information LC consisting of a unary predicate, with the name of the concept as the name of the predicate. For example (see Figure 7.4), “transverse\_reinforcement(Transverse\_reinforcement101)” is a unary predicate that represents an instance of the concept “transverse reinforcement” and “spacing(Spacing103)” is a unary predicate that represents an instance of the concept “spacing.” A relation fact is represented by a design information LC consisting of a binary or n-ary predicate, with the name of the relation as the name of the predicate. For example, “has(Transverse\_reinforcement101, Spacing103)” is a binary predicate that represents the relation that “Transverse\_reinforcement101” has a “Spacing103” and “has\_quantity(Spacing103, 6, Inches)” is a n-ary predicate which indicates that the quantity for “Spacing103” is 6 inches. Similar to regulatory information LCs, the recognition, extraction, and transformation of the concepts and relations into predicates are automatically conducted during the preceding information extraction and information transformation processes.

## 7.2.4 Functional Built-in Logic Clauses

Six types of functional built-in LCs were developed and included in the IRep and CRes schema, as per Table 7.1: unit conversion LCs, quantity comparison LCs, quantity conversion LCs, sum of quantities LCs, quantity arithmetic computation LCs, and rule checking LCs.

Table 7.1 Functional Built-in Logic Clauses

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

## 7.3 Software Implementation

### 7.3.1 Logic Programming Language

The proposed IRep and CRes schema was implemented in B-Prolog logic programming language. A FOL-based programming language is needed for representation to allow for automated reasoning. B-Prolog is a Prolog system with extensions for programming concurrency, constraints, and interactive graphics. It has a bi-directional interface with C and Java (Zhou 2012). Prolog is a logic platform that is based on HC representation and reasoning; it uses a “near-HC format” that allows clauses having two or more positive literals (e.g.,

“ $\neg B1 \wedge B2 \supset H$ ,” where B1, B2, and H are predicates). Although B-Prolog was selected in this dissertation, any other FOL-based programming language could be selected to represent the IRep and CRes schema instead; the proposed schema does not rely on any specific FOL-based programming language.

B-Prolog is a good fit for representing the IRep and CRes schema because: (1) B-Prolog builds in classic Prolog, which is the most widely-used logic programming language and reasoner (Costa 2009); (2) the built-in classic Prolog in B-Prolog has an underpinning reasoner that enables automated inference-making through well-developed unification, backtracking, depth-first search, and rewriting techniques (Portoraro 2011); and (3) the compatibility of B-Prolog with C and Java programming languages renders further ACC system user interface development and implementation smoother. The syntax in B-Prolog differs from the original FOL syntax, as summarized in Table 7.2. When another logic programming language is used, such as Answer Set Programming (ASP) or Datalog, the syntax of some functions may need to be adjusted. The slight difference in reasoning implementations across different FOL-based programming languages may also cause certain advantages or limitations in the reasoning. The discussion of the potential advantages and limitations of the different FOL-based programming languages is outside the scope of this dissertation.

Table 7.2 Syntax of FOL and B-Prolog

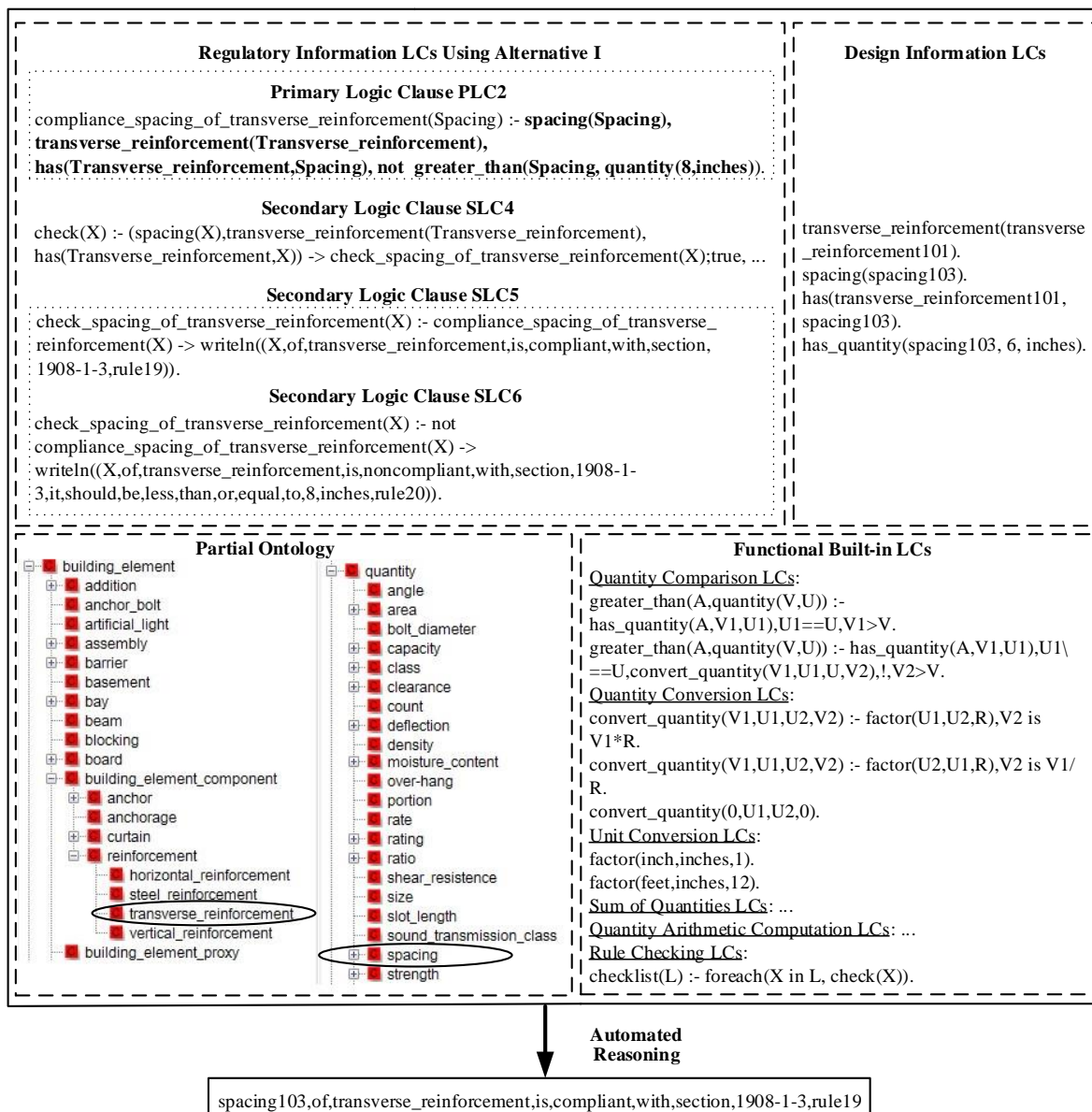
Element	Syntax in FOL	Syntax in B-Prolog
Conjunction	$\wedge$	,
Disjunction	$\vee$	;
Negation	$\neg$	not
Implication	$\supset$	:-
Constant	String starting with an upper-case letter	String starting with a lower-case letter
Variable	String starting with a lower-case letter	String starting with an upper-case letter
Universal quantifier	$\forall$	-
Existential quantifier	$\exists$	-
Predicate	pred(arg1,arg2,...)	pred(arg1,arg2,...)
Function	fun(arg1,arg2,...)	fun(arg1,arg2,...)
Rule	pred1(arg1,arg2,...) $\wedge$ pred2(arg1,arg2,...) ... $\wedge$ predn(arg1,arg2,...) $\supset$ predh(arg1,arg2,...)	predh(arg1,arg2,...) :- pred1(arg1,arg2,...), pred2(arg1,arg2,...)...., predn(arg1,arg2,...).
Fact	pred(arg1,arg2,...)	pred(arg1,arg2,...).
Directive	-	:- pred1(arg1,arg2,...), pred2(arg1,arg2,...)...., predn(arg1,arg2,...).

### 7.3.2 Regulatory Information Logic Clauses

#### 7.3.2.1 Alternative I

In Alternative I, regulatory information LCs (represented in the schema in the form of logic rules) are implemented as B-Prolog rules. The built-in “writeln()” predicate in B-Prolog is used for the output function. For executing the regulatory LCs, the user specifies the list of subjects (e.g., building elements such as walls and doors) or subjects and attributes to check and accordingly the subjects in the specified list are sequentially checked one by one. By default, a “select all” option is used if a user does not desire to specify specific subjects to

check. The sequence of checking in Alternative I is, thus, called subject-oriented. In the implementation of Alternative I, the search strategy is defined as follows: “for each selected subject instance, search through all regulatory information LCs to check if the activation conditions are satisfied, and if satisfied, then check the instance against the matched regulatory information LC.” The reasoning is supported by functional built-in LCs in the support module. An example of the implementation, corresponding to the example in Figure 7.4, is shown in Figure 7.5.



Note: “!” is the cut operator in B-Prolog which prevents a goal from being backtracked, “is” is the assignment operator in B-Prolog, and “->” is the symbol for representing implication within the body of a rule.

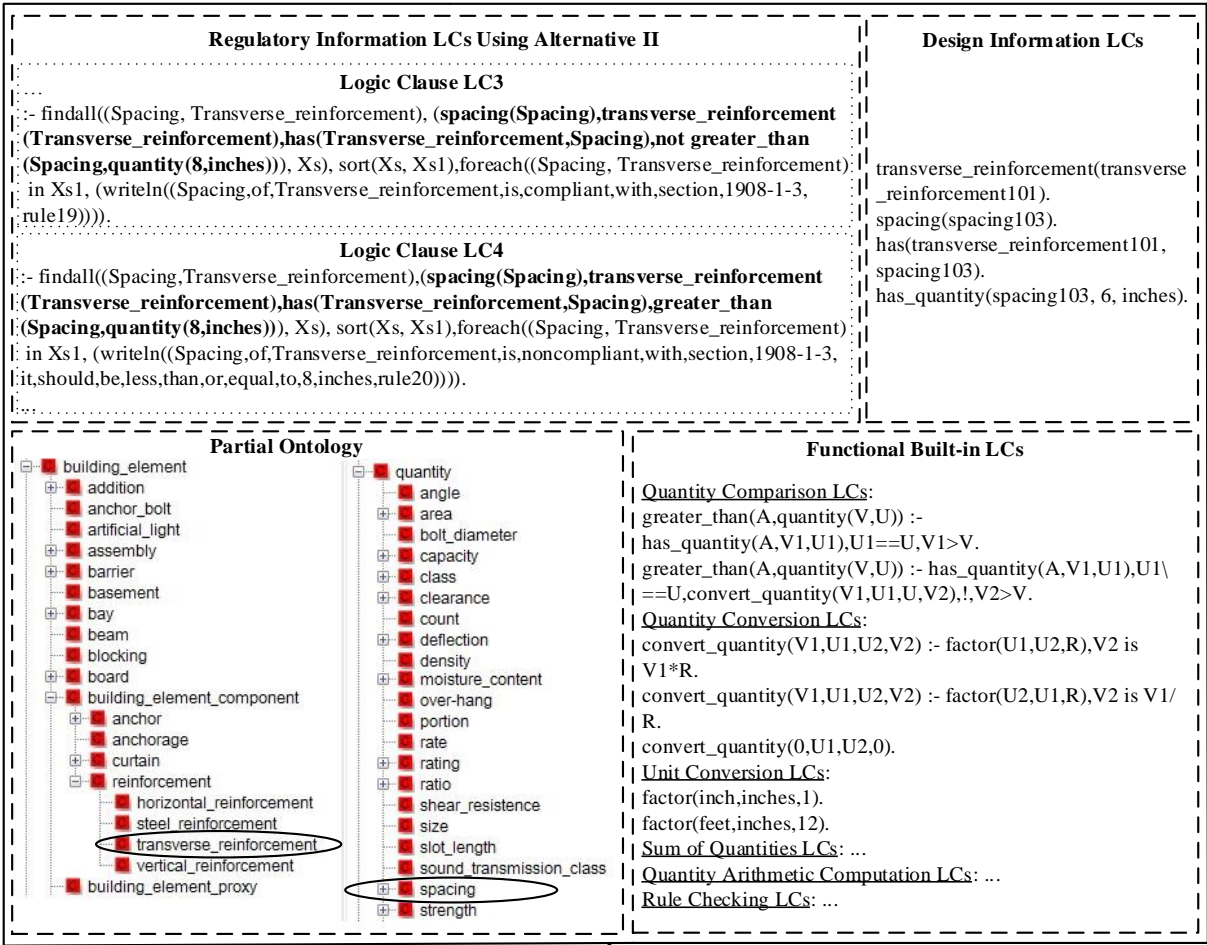
Figure 7.5 Alternative I of the Proposed IRep and CRes Schema (in B-Prolog Language)

### 7.3.2.2 Alternative II

In Alternative II, regulatory information LCs (represented in the schema in the form of logic rules) are implemented as B-Prolog directives. In comparison to B-Prolog rules, B-Prolog directives execute upon loading without conditions. B-Prolog directives were used instead of B-Prolog rules to allow for the execution of LCs upon loading, since activation conditions are



not used in Alternative II. In each directive, (1) the built-in “findall” predicate is used to leverage the inherent depth-first search strategy and backtracking techniques of B-Prolog to find all instances of the subject that satisfy the premise conditions of the requirement in the directive; (2) the “sort” predicate is used to sort the matched instances and remove duplicated instances; and (3) the “foreach” predicate is used to report the output results for each matched instance. In contrast to Alternative I, for executing the regulatory LCs in Alternative II, the user does not specify what subjects to check. All subjects that satisfy premise conditions in the regulatory information LCs are detected and checked. The sequence of checking follows the sequence of regulatory information LCs (i.e., the directives), which in turn follows the sequence of regulatory provisions in the original regulatory document. The sequence of checking in Alternative II is, thus, called regulation-oriented. An example of the implementation, corresponding to the example in Figure 7.4, is shown in Figure 7.6.



Automated Reasoning

spacing103,of,transverse\_reinforcement101,is,compliant,with,section,1908-1-3,rule19

Note: “!” is the cut operator in B-Prolog which prevents a goal from being backtracked, “is” is the assignment operator in B-Prolog, and “->” is the symbol for representing implication within the body of a rule.

Figure 7.6 Alternative II of the Proposed IRep and CRes Schema (in B-Prolog Language)

### 7.3.3 Design Information Logic Clauses

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

### 7.3.4 Functional Built-in Logic Clauses

The six types of functional built-in LCs in the IRep and CRes schema were implemented in B-Prolog syntax, as shown in Figure 7.5 and Figure 7.6.

One single rule checking LC is used in Alternative I and no rule checking LCs are used in Alternative II [not needed since the checking is initiated in each directive utilizing the inherent (“findall”) search strategies in B-Prolog]. As shown in Figure 7.5, the rule checking LC in Alternative I is: “*checklist(L) :- foreach(X in L, check(X)).*” This rule checking LC initiates the checking of subjects (in the user-specified list or default “select all” list), sequentially, one by one following the sequence in the list. In total, 71 functional built-in LCs were developed and used for Alternative I, and all 71 LCs except one (the rule checking LC) were used for Alternative II.

## 7.4 Experimental Testing and Evaluation

To empirically test the proposed IRep and CRes schema, Alternative I and Alternative II were tested in representing and reasoning about the quantitative regulatory requirements in Chapter 19 of IBC 2009 and the design information of the BIM test case (discussed in Section 6.3.1) for checking the compliance of the design. The results of noncompliance detection under each subschema alternative were evaluated in terms of precision and recall. To highlight the potential advantages of ACC using the proposed schema, the time efficiency of automated checking was also empirically tested.

## 7.4.1 Testing of Compliance Reasoning

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

### 7.4.1.1 Evaluation Metrics

The evaluation of compliance reasoning is conducted by comparing the noncompliance detection results with those in a manually-developed gold standard. The gold standard includes all noncompliance instances manually detected by the author. The evaluation was conducted using the following measures: precision, recall, and F1-measure. Precision, here, is the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that have been detected. Recall, here, is the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that should be detected. F1-measure is the harmonic mean of precision and recall.

### 7.4.1.2 Testing Using Perfect Information

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

For testing Alternative 1, both regulatory information LCs and design information LCs were

manually represented/coded based on Gold Standard I (i.e., the gold standard of Alternative I). Gold Standard I was composed of two subparts: (1) the gold standard of regulatory information LCs in Chapter 19 of IBC 2009 under Alternative I, which included 264 LCs (in the form of B-Prolog rules), consisting of 66 primary LCs and 198 secondary LCs (i.e., three secondary LCs for each primary LC), and (2) the gold standard of design information LCs in the BIM test case, which included 1,442 LCs (in the form of B-Prolog facts). For example, Figure 7.5 shows the gold standard for representing provision *RP2* and a set of design information, where PLC2 is one of the 264 LCs and “spacing(spacing103)” is one of the 1,442 LCs. The reasoning was then conducted automatically using the B-Prolog reasoner.

For testing Alternative II, the same testing procedure was followed, except that both regulatory information LCs and design information LCs were manually coded based on Gold Standard II (i.e., the gold standard of Alternative II). Gold Standard II was composed of two subparts: (1) the gold standard of regulatory information LCs in Chapter 19 of IBC 2009 under Alternative II, which included 137 LCs (in the form of B-Prolog directives), and (2) the gold standard of design information LCs in the BIM test case, which included 1,442 LCs (in the form of B-Prolog facts). For example, Figure 7.6 shows the gold standard for representing provision *RP2* and a set of design information, where LC3 is one of the 137 LCs and “spacing(spacing103)” is one of the 1,442 LCs.

#### 7.4.1.3 Testing Using Imperfect Information

The testing using imperfect information was conducted using a similar procedure to that of

testing using perfect information, except that a set of automatically-coded regulatory information LCs were used instead of the manually-coded ones. These automatically-coded LCs were automatically generated from Chapter 19 of IBC 2009 using algorithms for automated information extraction (to automatically extract information from regulatory documents into semantic tuples, as presented in Chapter 3) and automated information transformation (to automatically transform the semantic tuples into LCs, as presented in Chapter 4). The use of automatically-coded regulatory information LCs allows for evaluating the performance of compliance reasoning using imperfect information (i.e., because the automatically-coded LCs contain errors). For both alternatives, Alternative I and Alternative II, 21 of the regulatory requirements contained errors. While the same information extraction algorithm was used for both Alternative I and Alternative II, the information transformation algorithm was slightly modified for Alternative II due to the differences in terms of regulatory information representation. In Alternative II, the transformation results are B-Prolog directives instead of B-Prolog rules.

#### **7.4.2 Testing of Time Performance**

To compare the time efficiency of the two alternative subschemas, the durations of automated compliance reasoning using perfect information, under Alternative I and Alternative II were calculated using the time keeping predicates in B-Prolog. Since Alternative I is subject-oriented while Alternative II is regulation-oriented, the duration of compliance reasoning is measured differently for each alternative. For Alternative I, the duration is

measured from the time of initializing the compliance reasoning about the first design fact to the time of finishing compliance reasoning about the last design fact (design information LC No. 1,442). For Alternative II, the duration is measured from the time of initializing compliance reasoning with the first regulatory requirement to the time of finishing compliance reasoning with the last regulatory requirement (regulatory information LC No. 137).

### **7.4.3 Results and Discussion**

#### **7.4.3.1 Results of Compliance Reasoning Performance**

##### **Results Using Perfect Information**

The experimental results are summarized in Table 7.3. When using perfect information, on the testing data, both Alternative I and Alternative II achieved 100% precision, recall, and F1-measure in noncompliance detection. This shows that the proposed IRep and CRes schema is effective in supporting ACC. The compliance checking results and suggestions for fixing noncompliance instances were also correctly reported in the output.

Table 7.3 Experimental Results

Subschema	Parameter/measure	Results	
Alternative I (closed world assumption)	Number of noncompliance instances in gold standard	79	79
	Number of noncompliance instances detected	79	89
	Number of noncompliance instances correctly detected	79	78
	Precision of noncompliance detection	100%	87.6%
	Recall of noncompliance detection	100%	98.7%
	F1-measure of noncompliance detection	100%	92.8%
Alternative II (open world assumption)	Number of noncompliance instances in gold standard	79	79
	Number of noncompliance instances detected	79	62
	Number of noncompliance instances correctly detected	79	61
	Precision of noncompliance detection	100%	98.4%
	Recall of noncompliance detection	100%	77.2%
	F1-measure of noncompliance detection	100%	86.5%

Figure 7.7 shows the checking results of “wall1” to “wall5” using Alternative I. For example, “wall1” has “height3,” “thickness1,” and “unbalanced\_fill1;” and “wall2” has “height4,” “thickness2,” and “unbalanced\_fill2,” where Rule43 and Rule44 focus on height checking, Rule43-1 and Rule45 focus on thickness checking, and Rule43-2 and Rule46 focus on unbalanced fill checking. Figure 7.8 shows the checking results of “wall1” to “wall5” using Alternative II, where Rule44, Rule 45, and Rule 46 represent the noncompliance cases of “height,” “thickness,” and “unbalanced fill,” respectively, and Rule 43 represents the compliance cases of all three regulatory requirements jointly.



```

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

```

Figure 7.7 Sample Compliance Checking Results Using Alternative I

```

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

```

Figure 7.8 Sample Compliance Checking Results Using Alternative II

### Results Using Imperfect Information

When using imperfect information, on the testing data, Alternative I and Alternative II achieved 87.6% (95% confidence interval [79.2%, 93.0%]), 98.7% (95% confidence interval [93.2%, 99.8%]), and 92.8% (95% confidence interval [85.6%, 96.3%]) and 98.4% (95% confidence interval [91.4%, 99.7%]), 77.2% (95% confidence interval [66.8%, 85.1%]), and 86.5% (95% confidence interval [77.2%, 91.8%]) precision, recall, and F1-measure in noncompliance detection, respectively. The recall of Alternative I outperformed that of Alternative II, while the precision of Alternative II outperformed that of Alternative I. This reflects the trade-off between precision and recall.

In Alternative I, a high recall is achieved because it can block some errors in information

extraction and information transformation from propagating to false negatives in noncompliance detection results; a total of 39 incorrect regulatory information LCs caused by errors in information extraction and information transformation occurred, yet only 1 of them propagated into a false negative in noncompliance detection. Errors in predicates other than quantity comparison predicates [e.g., `greater_than(Spacing,quantity(8,inches))` in Figure 7.5] could be blocked from leading to false negatives. Because, in Alternative I, all selected design subjects are checked, noncompliance instances are less likely to be missed. However, most of these information extraction and information transformation errors still lead to false positives, which makes the precision relatively lower than recall.

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

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

#### 7.4.3.2 Results of Time Performance

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

## **8 CHAPTER 8 – PROTOTYPE SYSTEM IMPLEMENTATION AND EXPERIMENTAL TESTING**

The proposed ACC method was implemented in a proof-of-concept prototype system named Semantic Natural Language Processing-based Automated Compliance Checking (SNACC) system. The SNACC system was implemented using Java programming language (Oracle 1999), General Architecture for Text Engineering (GATE) tools (Cunningham et al. 2012), Python programming language (Python 2.7.3), B-Prolog logic programming platform and reasoner (Zhou 2012), and Java Standard Data Access Interface (JSDAI) tools.

### **8.1 System Architecture**

The system architecture is illustrated in Figure 8.1. It is composed of three main modules: (1) regulatory information extraction and transformation module, (2) design information extraction and transformation module, and (3) compliance reasoning module. The regulatory information extraction and transformation module is composed of regulatory information extraction and regulatory information transformation submodules. At the core of the regulatory information extraction submodule is the information extraction algorithm, including the information extraction rules and the conflict resolution rules. The syntactic features in the patterns of the rules are generated using GATE's Processing Resources (e.g., tokenizer), while the semantic features are generated from an ontology using GATE's Processing Resources (e.g., gazetteer). The information extraction algorithm interacts with the Processing Resources using GATE's application program interface (API) in Java.

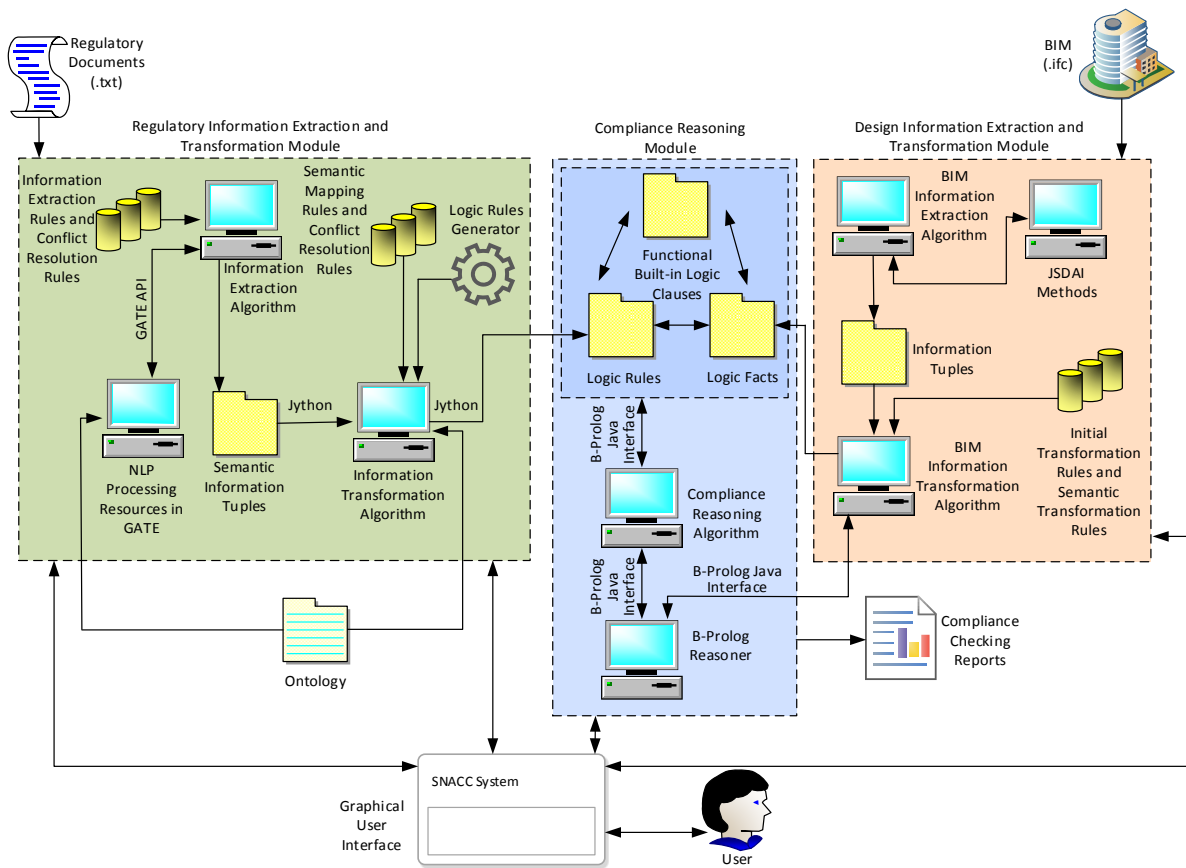


Figure 8.1 System Architecture of the SNACC System

At the core of the regulatory information transformation submodule is the information transformation algorithm, including the semantic mapping rules and the conflict resolution rules. The syntactic and semantic features in the patterns of the rules are extracted from the information tags of the semantic information tuples. The information transformation algorithm interacts with the other modules of the SNACC system (in Java) through Jython.

The design information extraction and transformation module is composed of the BIM information extraction and the BIM information transformation submodules. At the core of the BIM information extraction submodule is the BIM information extraction algorithm,

including the entity and attribute extraction rules. The data types of the entities and attributes in the conditions of the rules are extracted from the BIM. At the core of the BIM information transformation submodule is the BIM information transformation algorithm, including the transformation rules (for initial transformation and alignment transformation). To execute the transformation rules for alignment transformation, the information transformation algorithm interacts with B-Prolog reasoner through B-Prolog's interface with Java.

At the core of the compliance reasoning module is the compliance reasoning algorithm, including the functional built-in logic clauses. The compliance reasoning algorithm controls and supports the reasoning about the logic rules and logic facts using the reasoner in B-Prolog. The compliance reasoning algorithm interacts with B-Prolog reasoner through the B-Prolog's interface with Java. A user interacts with all the three modules through a graphical user interface.

## **8.2 System Implementation**

The main platform of the SNACC system was built using Java programming language (Java Standard Edition Development Kit 6u45). The regulatory information extraction algorithm was implemented using the GATE's Processing Resources and Java programs. The following Processing Resources were used: (1) Java Annotation Patterns Engine (JAPE) rules for encoding the information extraction rules; (2) the English Tokenizer, Sentence Splitter, POS Tagger, and Gazetteer in the A Nearly-New Information Extraction (ANNIE) system for

tokenization, sentence splitting, POS tagging, and gazetteer compiling; (3) the Morphological Analyzer for morphological analysis; and (4) the Flexible Gazetteer for generating semantic features based on the ontology. The conflict resolution rules were coded as Java conditional statements. The information extraction algorithm interacts with the Processing Resources using GATE's API 7.0.

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

The BIM information extraction and transformation algorithm was implemented in Java programs and B-Prolog rules. The Java Standard Data Access Interface (JSDAI) runtime (JSDAI4.1.505.v201112201320) was used to access the information in IFC-based BIMs (.ifc files) for entity and attribute extraction. String processing methods in Java were used for initial transformation. Static rules and dynamic rules in B-Prolog were used for alignment transformation. The rules for entity extraction, attribute extraction, and initial transformation were coded as Java conditional statements. The rules for alignment transformation (i.e., SeTr rules) were coded as B-Prolog rules.

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

The graphical user interface of the SNACC system is shown in Figure 8.2 to Figure 8.10. The SNACC system requires the download of the GATE tool and the availability of a building ontology to execute the regulatory information extraction and transformation algorithms. As the first two steps of executing the SNACC system, an installed GATE and a building ontology must be specified (Figure 8.2, Figure 8.3).

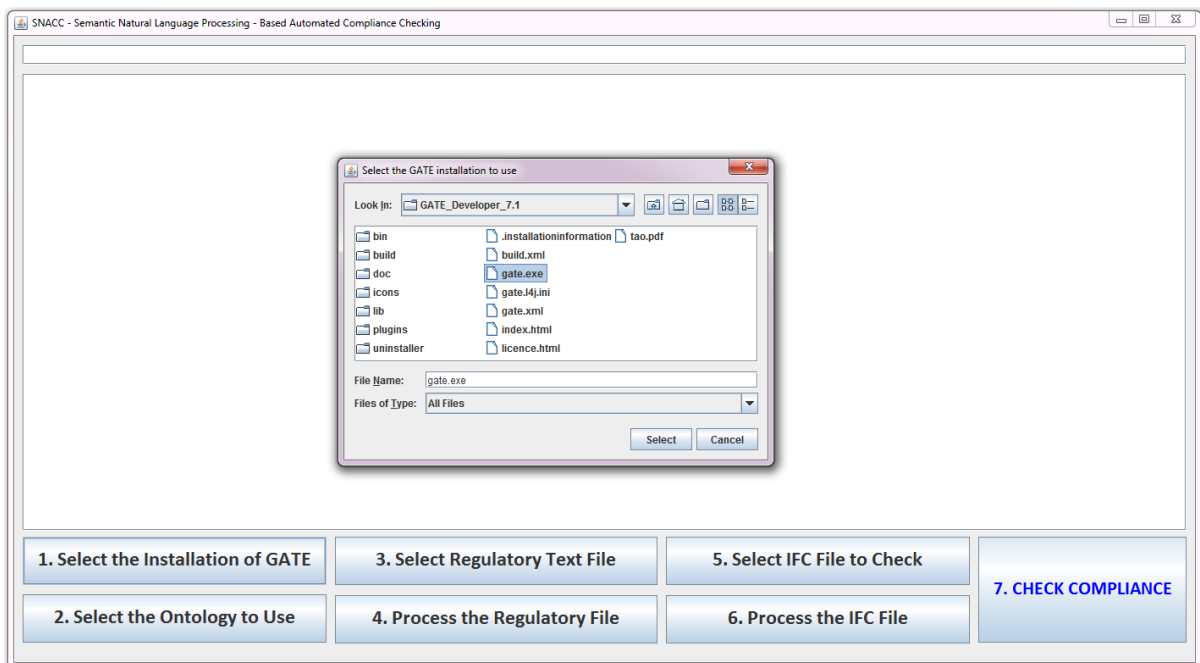


Figure 8.2 Graphical User Interface of the SNACC System for GATE Selection



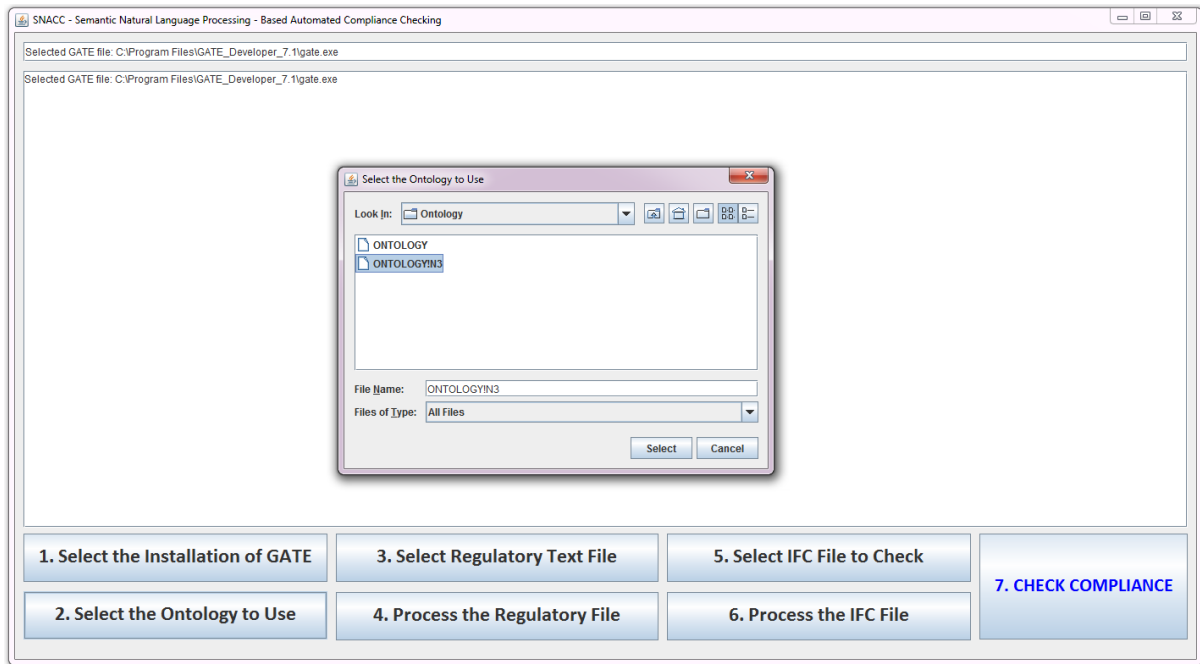


Figure 8.3 Graphical User Interface of the SNACC System for Ontology Selection

A user could then select the regulatory document (.txt file) (Figure 8.4) and the BIM (.ifc file) (Figure 8.5) for automated compliance checking. The information extraction and information transformation algorithms for regulatory information (Figure 8.6 and Figure 8.7) and design information (Figure 8.8 and Figure 8.9) could be executed in parallel. After all information have been extracted and transformed, pressing the “check compliance” button activates the automated reasoning process using B-Prolog (Figure 8.10). The compliance checking results are then automatically displayed to the user in the text field of the graphical user interface (as shown in Figure 8.10).

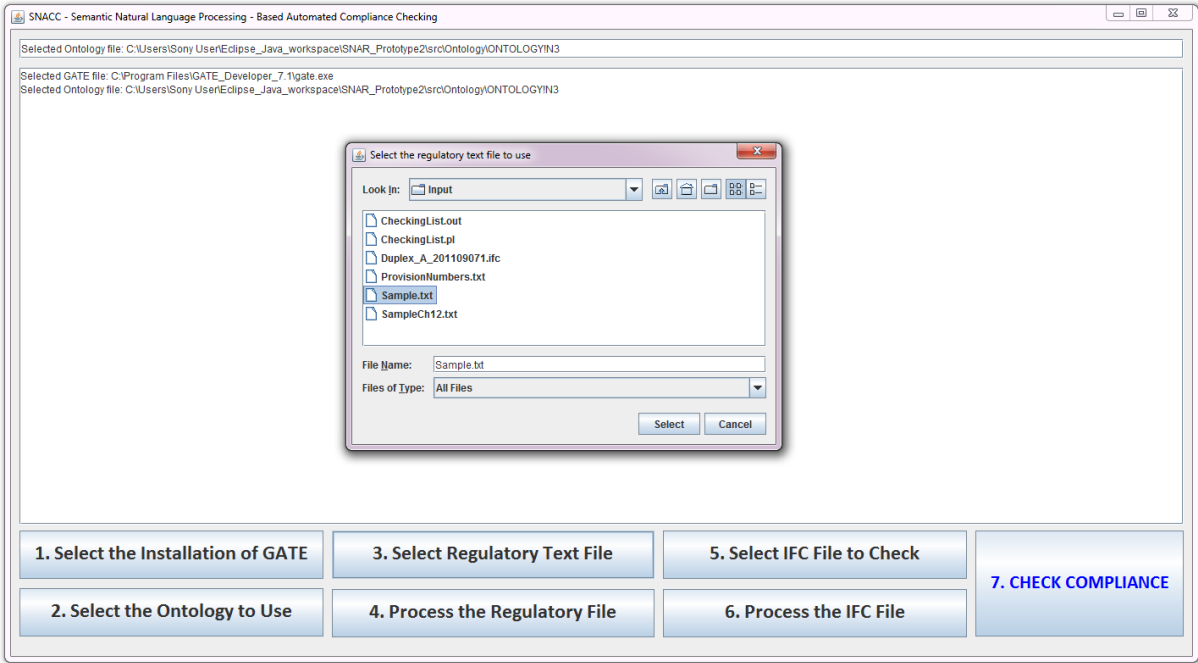


Figure 8.4 Graphical User Interface of the SNACC System for Regulatory Text Selection

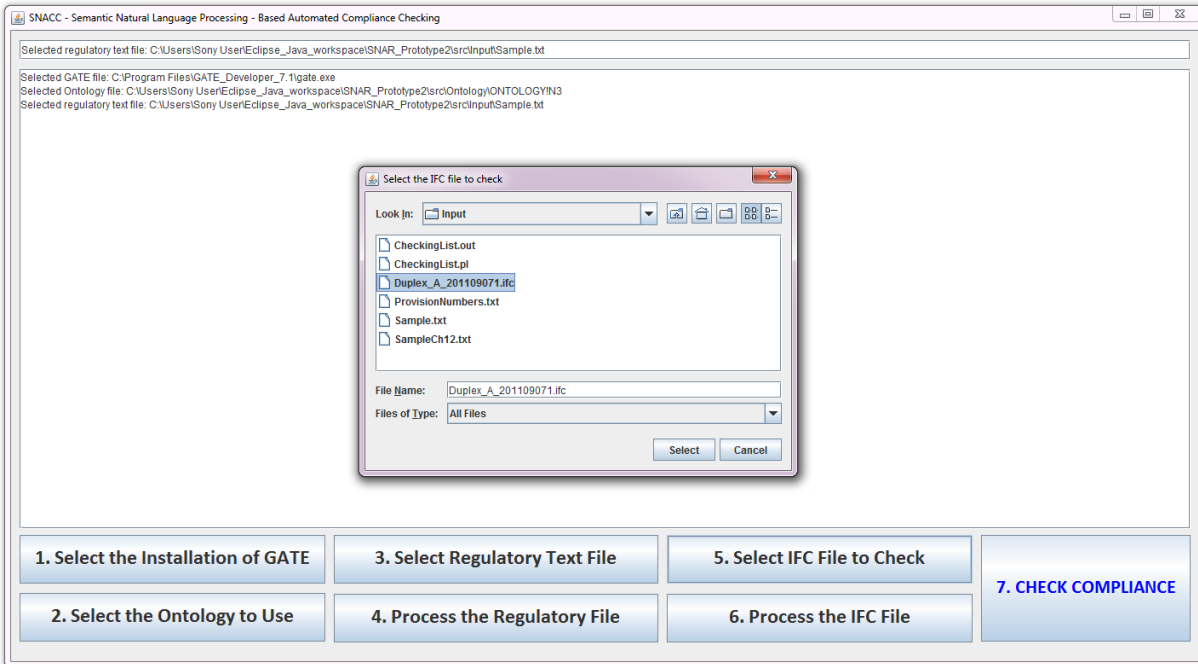


Figure 8.5 Graphical User Interface of the SNACC System for .ifc File Selection

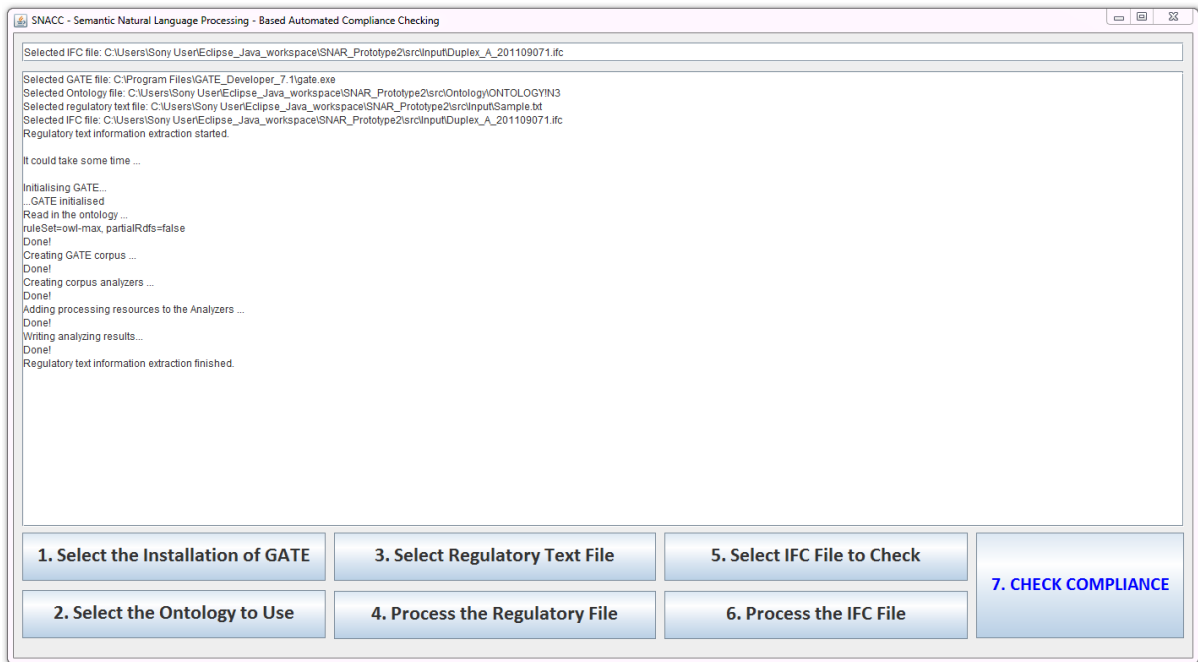


Figure 8.6 Graphical User Interface of the SNACC System for Regulatory Information Processing (a)

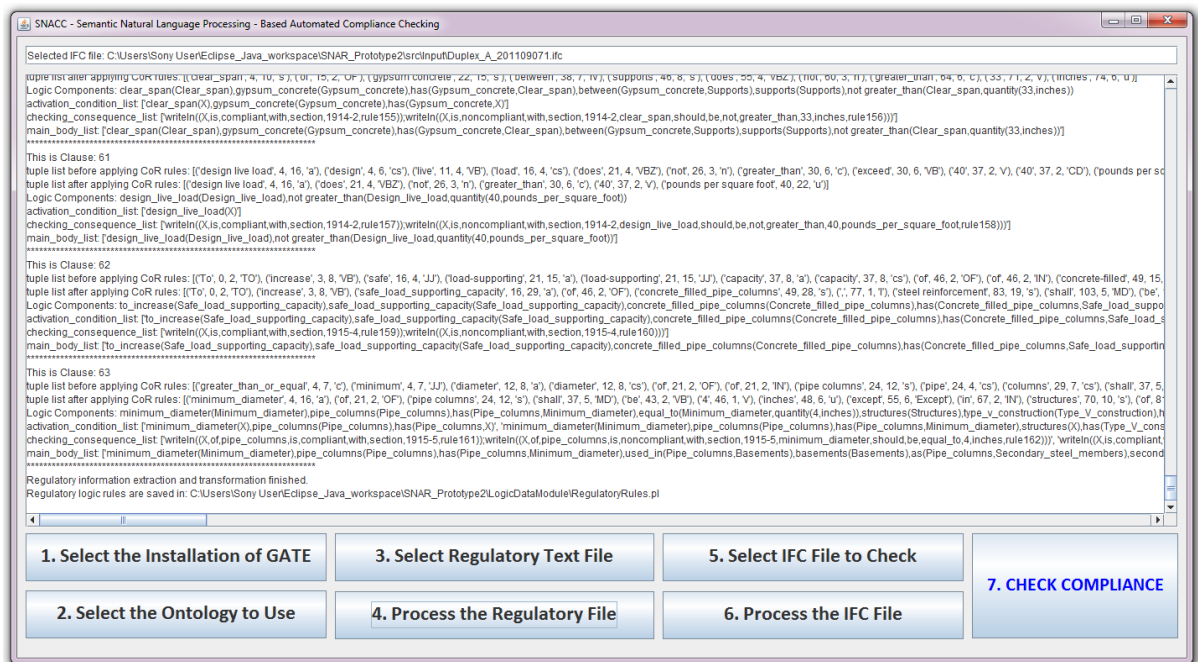


Figure 8.7 Graphical User Interface of the SNACC System for Regulatory Information Processing (b)

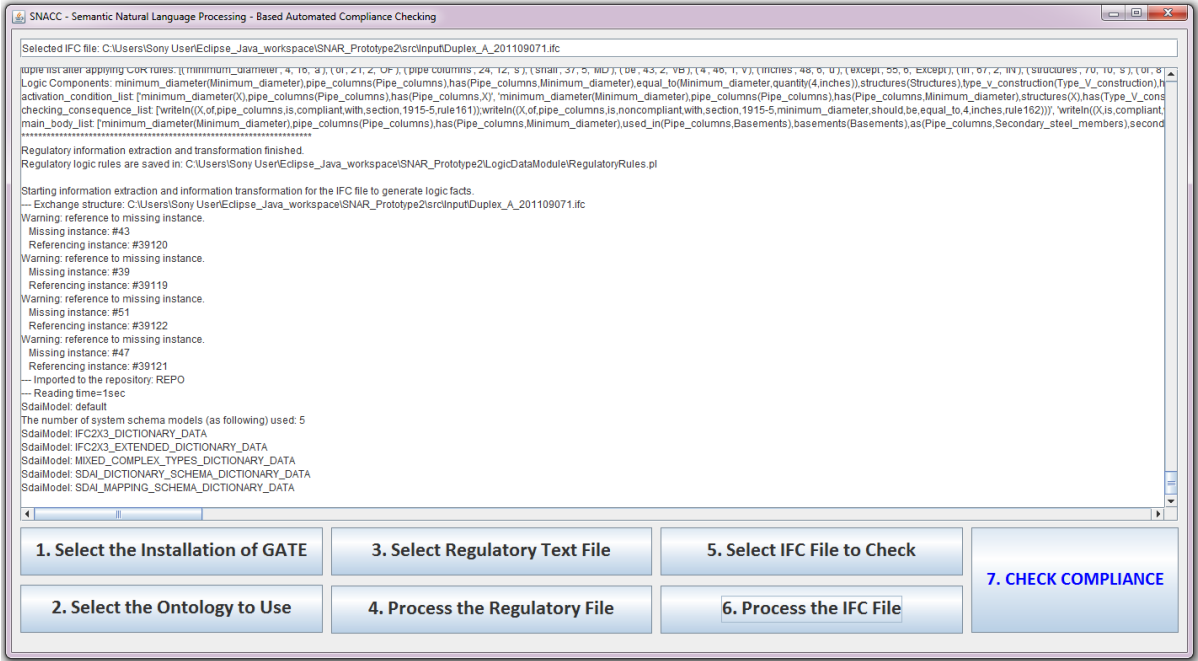


Figure 8.8 Graphical User Interface of the SNACC System for Design Information Processing (a)

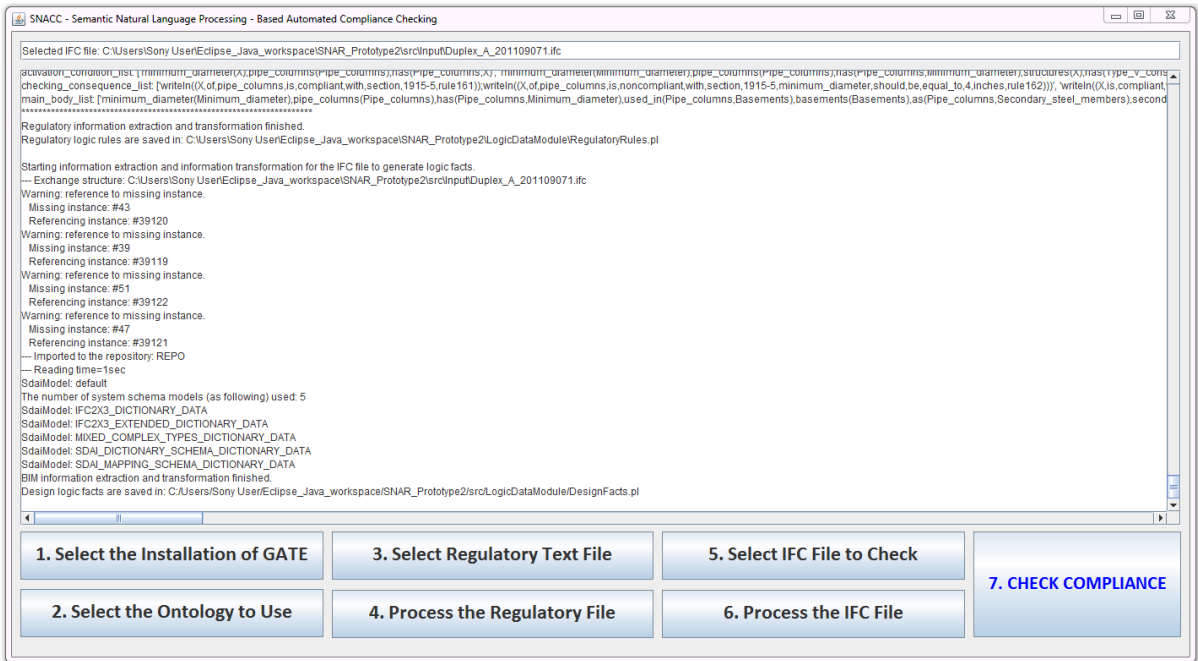


Figure 8.9 Graphical User Interface of the SNACC System for Design Information Processing (b)

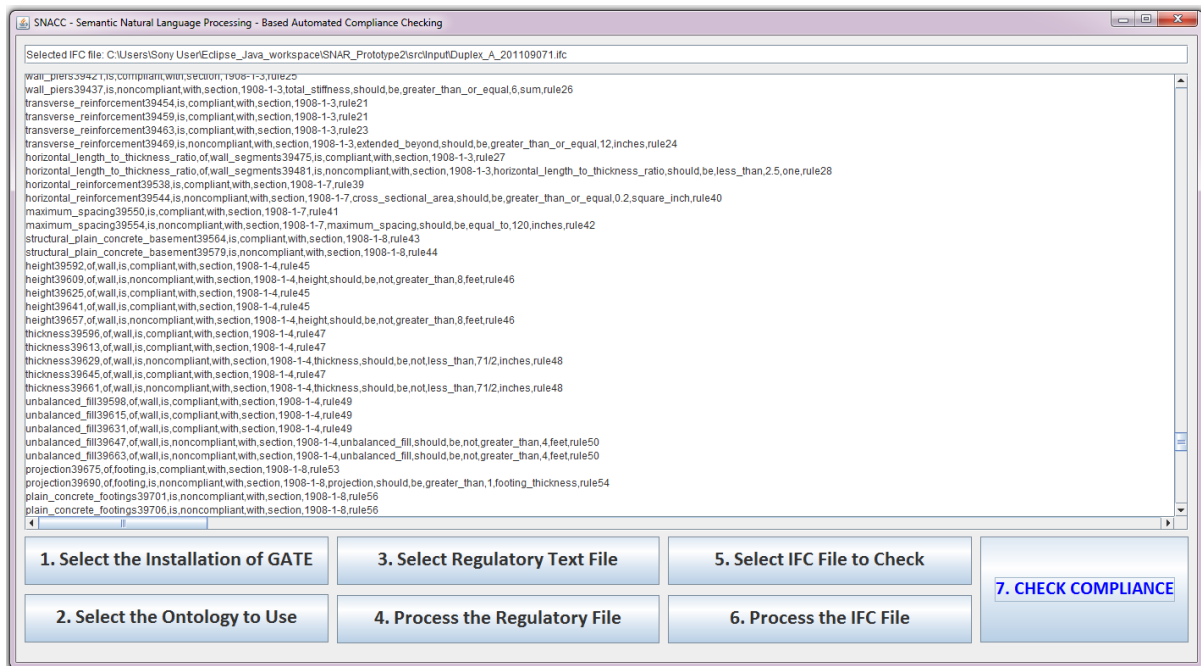


Figure 8.10 Graphical User Interface of the SNACC System for Automated Compliance Checking

### 8.3 Data for System Testing

The SNACC system was tested in checking the compliance of the BIM test case with Chapter 19 of IBC 2009. IBC was selected because it is predominantly adopted in the United States. Chapter 19 of IBC 2009 was then randomly selected. For the test case, it was developed based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences (East 2013). Design information were added in the BIM model, based on the extended IFC schema. The test case included design information for each provision in Chapter 19 of IBC 2009. The design information included both compliant and noncompliant design information. If a provision has more than one requirement, then compliant and noncompliant design information for each requirement is included. For

example, the following regulatory provision (*RPI*) is a complex provision that contains three quantitative requirements: “*In dwellings assigned to Seismic Design Category D or E, the height of the wall shall not exceed 8 feet (2438 mm), the thickness shall not be less than 7 1/2 inches (190 mm), and the wall shall retain no more than 4 feet (1219 mm) of unbalanced fill*” (*Provision 1908.1.8 of IBC 2009*). Thus, five information sets were created for *RPI* which correspond to the scenarios that (1) only height is noncompliant, (2) only thickness is noncompliant, (3) only unbalanced fill is noncompliant, (4) all three attributes are noncompliant, and (5) no attributes are noncompliant.

#### **8.4 Evaluation Metrics**

The ACC prototype system was evaluated using precision, recall, and F1-measure of noncompliance detection. Precision, here, is defined as the number of correctly detected noncompliance instances divided by the total number of noncompliance instances detected. Recall, here, is defined as the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that should be detected. F1-measure is the harmonic mean of precision and recall. A manually-developed gold standard was used for the evaluation. A gold standard refers to a benchmark against which testing results are compared for evaluation. The gold standard includes the ground truth of compliant and noncompliant instances. For example, Figure 8.11 shows the ground truth of a “thickness” instance for an exterior basement wall in the gold standard (checked for compliance with *Provision 1909.6.1 of IBC 2009*).

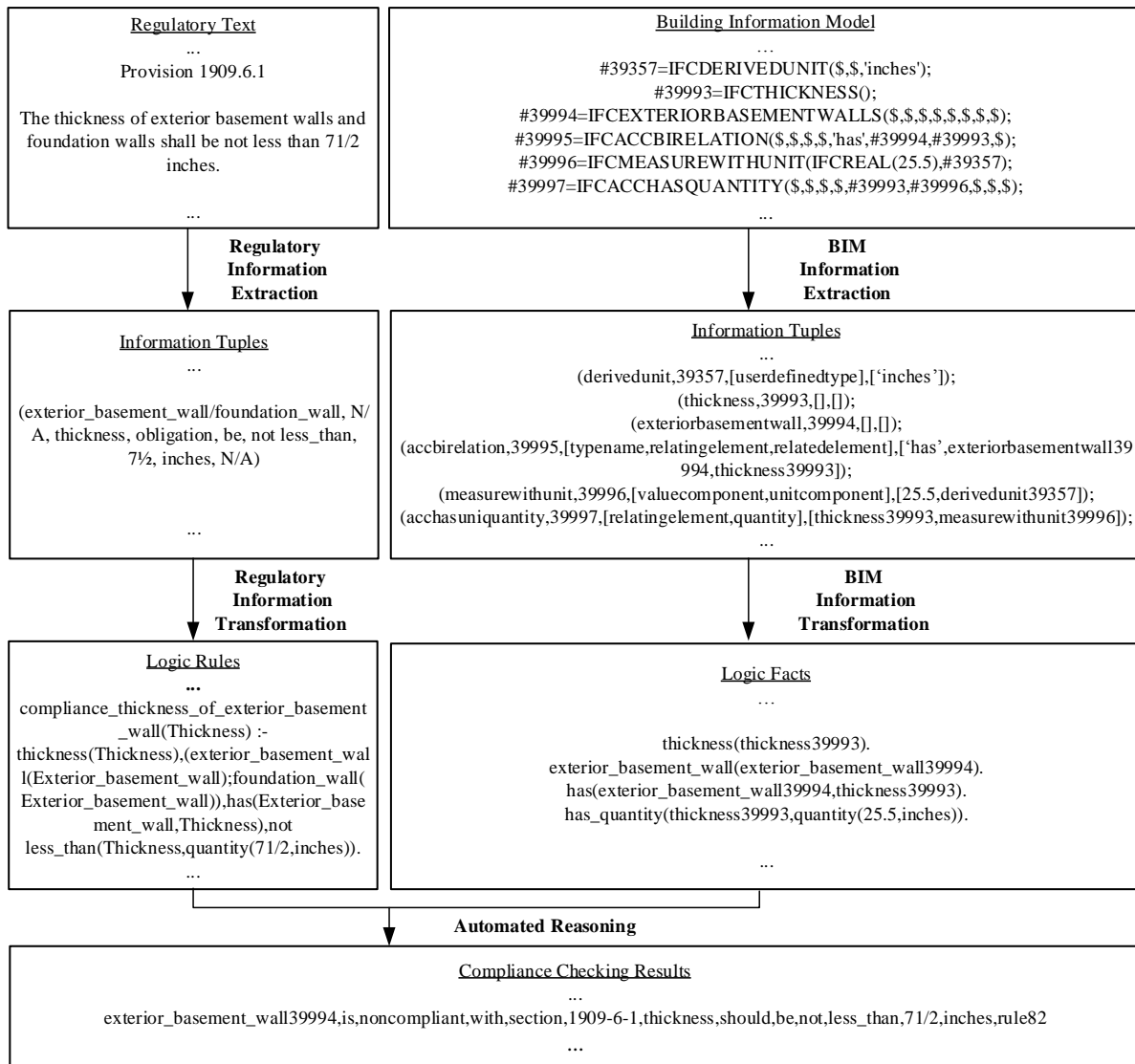


Figure 8.11 Gold Standard of an Example Provision and Design Information

## 8.5 Results and Discussion

The testing results are summarized in Table 8.1. As shown in Table 8.1, the precision, recall, and F1-measure of noncompliance detection is 87.6% (95% confidence interval [79.2%, 93.0%]), 98.7% (95% confidence interval [93.2%, 99.8%]), and 92.8% (95% confidence interval [85.6%, 96.3%]), respectively. The relevant provision numbers and rule ID numbers

for the compliant and noncompliant instances were also correctly reported. For each noncompliance instance, a suggestion on how to fix the noncompliance case was also correctly reported (partially shown in Figure 8.10).

Table 8.1 Noncompliance Detection Testing Results

<b>Parameter/measure</b>	<b>Result</b>
Number of noncompliance instances in gold standard	79
Number of noncompliance instances detected	89
Number of noncompliance instances correctly detected	78
Recall of noncompliance detection	98.7%
Precision of noncompliance detection	87.6%
F1-measure of noncompliance detection	92.8%

These high performance results show that the proposed ACC system is promising. In addition, the fact that the proposed ACC system achieved higher recall (98.7%) than precision (87.6%) shows its suitability for the ACC application; in noncompliance detection, recall is more important than precision. Recall errors are more critical because they might result in missing noncompliance instances, whereas precision errors could be easily double-checked and filtered out by the user. The system achieved a high recall for two main reasons: (1) the information representation schema for automated reasoning follows the closed world assumption and therefore blocks certain errors in information extraction and information transformation from propagating into noncompliance detection results; and (2) the subjects (e.g., building elements) for checking are specified in a list and sequentially checked one by one, which ensures that all subjects in the list are checked for compliance and avoids false negatives.



An error analysis was also conducted to identify the sources of the errors in noncompliance detection. The noncompliance detection errors originated from errors in regulatory information extraction and regulatory information transformation; there were no errors in BIM information extraction, BIM information transformation, and compliance reasoning. Five main sources of errors are recognized: (1) limitations of GATE's Processing Resources: for example, the Flexible Gazetteer in GATE failed to recognize the term "bar" although it exists in the ontology; (2) limitations of conflict resolution rules in regulatory information extraction: for example, one conflict resolution rule states that if no "compliance checking attribute" was extracted and extra "subject" candidates were extracted, then place the "subject" candidate that is closest to the "quantity value" as the attribute. This rule led to an incorrect extraction of "clear height" as the compliance checking attribute instance in the sentence "Transverse reinforcement shall be extended beyond the pier clear height for at least 12 inches" (Provision 1908.1.3 of IBC 2009); (3) missing patterns in information extraction rules in regulatory information extraction: for example, the pattern for "the percentages of the total weight of cementitious materials" was missing in the information extraction rules for extracting quantity reference. Therefore, "the percentages of the total weight of cementitious materials" was not extracted as a quantity reference from the following part of sentence: "the maximum weight of fly ash, other pozzolans, silica fume or slag that is included in the concrete shall not exceed the percentages of the total weight of cementitious materials permitted by ACI 318, Section 4.4.2" (Provision 1904.4.2 of IBC 2009); (4) structural

ambiguity caused by conjunctive terms in regulatory information transformation: for example, in the following part of sentence, there are two possible syntactic uses of “or,” either linking “structural shapes” and “pipe” or linking “structural shapes” and “pipe embedded in the concrete core with sufficient clearance”: “structural shapes or pipe embedded in the concrete core with sufficient clearance...” (Provision 1915.4 of IBC 2009). The ability of the semantic mapping rules to handle structural ambiguity is limited by the development text, which may lead to errors in regulatory information transformation; and (5) limitations of certain semantic mapping rules in regulatory information transformation: for example, a semantic mapping rule selects the immediate left neighbor of a preposition as the first argument of that preposition. In cases in which the immediate left neighbor of a preposition is not its real first argument, this semantic mapping rule causes errors. For example, in the following part of sentence, “approved agency” was mistakenly identified as the first argument for the preposition “on” rather than “strength tests”: “strength tests for shotcrete shall be made by an approved agency on specimens...” (Provision 1913.10 of IBC 2009).

## **9 CHAPTER 9 – CONCLUSIONS, CONTRIBUTIONS, LIMITATIONS, AND RECOMMENDATIONS FOR FUTURE RESEARCH**

### **9.1 Conclusions**

#### **9.1.1 Conclusions for the Proposed Regulatory Information Extraction Method and Algorithm**

A semantic, rule-based NLP method and algorithm for automatically extracting regulatory information from building codes for supporting ACC in the construction domain was developed. A set of pattern-matching-based IE rules and CR rules are used in information extraction. The patterns are represented in terms of syntactic and semantic text features. NLP techniques are utilized to capture the syntactic features of the text, and a domain ontology is used to capture the semantic ones. PSG-based phrasal tags are used in syntactic analysis to reduce the number of needed patterns. Information elements are extracted separately and sequentially to further limit the number of needed patterns. The information extraction is relatively deep; it aims to achieve full sentence analysis to extract all information of a requirement for further representation in a logic-based rule format.

The proposed algorithm was tested in extracting quantitative requirements from IBC 2009. A comparison of the extracted information element instances with those in a semiautomatically developed gold standard showed an average precision, recall, and F1-measure of 96.9%, 94.4%, and 95.6% respectively. A preliminary reusability testing of the IE and CR rules on a

randomly selected piece of text from the Internet also achieved an F1-measure greater than 90%, which shows the good potential of reusability of the rules. These high performance results indicate that the proposed information extraction approach is promising. An error analysis also pinpointed the sources of errors in the experimental results and identified potential solutions for the possibility of further performance enhancement.

### **9.1.2 Conclusions for the Proposed Regulatory Information Transformation Method and Algorithm**

A semantic, rule-based method and algorithm for automatically transforming the extracted regulatory information into logic rules for supporting ACC in the construction domain was developed. A set of semantic mapping (SM) rules and conflict resolution rules (CR) are used in information transformation. CR rules resolve conflicts between information instances, while SM rules transform the information instances into logic clause elements. The SM rules use context-aware and flexible information patterns. Both syntactic and semantic information tags are utilized in the patterns. Syntactic information tags (e.g., POS tags) are generated using NLP techniques. A semantic model helps recognize the semantic information tags of each extracted information instance. A “consume and generate” mechanism was proposed to handle complex sentence components and execute the SM rules. The information transformation method, thus, processes almost all terms of a sentence. Such full sentence processing enables deep NLP towards natural language understanding.

The proposed information transformation algorithm was tested in transforming information instances of quantitative requirements, which were automatically extracted from Chapter 19 of IBC 2009, into logic clauses. The transformation results were compared with a manually-developed gold-standard. The results showed 98.2%, 99.1%, and 98.6% precision, recall, and F1-measure, respectively. This high performance shows that the proposed information transformation method is promising. An error analysis also pinpointed the sources of errors in the experimental results and identified potential solutions for the possibility of further performance enhancement.

### **9.1.3 Conclusions for the Proposed IFC Extension Method and Algorithm**

A semantic, rule-based and machine learning-based method and algorithm for semiautomatically extending the IFC schema with concepts in building codes for supporting ACC in the construction domain was developed. The proposed method utilizes semantic natural language processing (NLP) techniques and machine learning techniques, and is composed of four primary methods that are combined into one computational platform: (1) a method for concept extraction that utilizes POS-pattern-matching-based rules to extract regulatory concepts from regulatory documents, (2) a method for identifying and selecting the most related IFC concepts to the extracted regulatory concept, which utilizes term-based and semantic-based matching algorithms to find candidate related IFC concepts and a semantic similarity (SS) scoring and ranking algorithm to measure the SS between each candidate IFC concept and a regulatory concept, (3) a machine learning classification method

for predicting the relationship between the extracted regulatory concepts and their most related IFC concepts based on the syntactic and semantic features of their terms, and (4) a mapping rule-based method for integrating the regulatory concepts into the IFC schema based on their relationship with IFC concepts.

The proposed IFC extension method was evaluated on extending the IFC schema with regulatory concepts from Chapter 19 of IBC 2009, using a manually-developed gold standard. Each of the four methods were evaluated separately, and achieved 91.7%, 84.5%, 87.94% and 100% F1-measure, adoption rate, precision, and F1-measure, respectively. The performance results indicate that the proposed IFC extension method is potentially effective. The results also show that semantic features of the concept terms and their interrelationships are helpful in IFC extension and result in performance improvement. An error analysis also pinpointed the sources of errors in the experimental results and identified potential solutions for the possibility of further performance enhancement.

#### **9.1.4 Conclusions for the Proposed BIM Information Extraction and Transformation**

##### **Method and Algorithm**

A semantic, rule-based method and algorithm for automatically extracting design information from IFC-based building information models (BIMs) and automatically transforming the extracted design information into logic facts for supporting ACC in the construction domain was developed. The proposed method utilizes late binding data access in Java Standard Data

Access Interface (JSDAI) to extract all entities and their attributes from an .ifc file into information tuples using their metadata at the EXPRESS model level (i.e., EXPRESS datatypes). Data type-based transformation rules are used to transform all extracted BIM information into logic facts, while static and dynamic logic rules are used to align the logic facts with regulatory information.

The proposed BIM information extraction and transformation method was evaluated using a BIM test case. The design information in the .ifc file of the test case was automatically extracted into semantic tuples and transformed into logic facts. The extracted information tuples and transformed logic facts were compared with those in a manually-developed gold standard, and the results were evaluated in terms of precision and recall. The results showed 100% precision and recall for both information extraction and information transformation. This indicates that the proposed method is effective and the flow of design information from .ifc files directly to logic format is feasible.

### **9.1.5 Conclusions for the Proposed Information Representation and Compliance**

#### **Reasoning Schema**

A logic-based information representation and compliance reasoning (IRep and CRes) schema for representing, both, regulatory information and design information to prepare for utilizing the inference-making capabilities of logic reasoners for supporting ACC in the construction domain was developed. The schema formalizes the representation of regulatory information

and design information in the form of semantic-based (ontology-based) logic clauses that could be directly used for automated compliance reasoning. The proposed schema was implemented in the B-Prolog platform. B-Prolog was selected to utilize its embedded classic Prolog logic programming language; its automated reasoning facilities such as search strategies, backtracking, and unification mechanisms; and its constraint solving capabilities. Two alternative subschemas, Alternative I and Alternative II, were proposed and tested, following a closed world assumption and an open world assumption in noncompliance detection, respectively. Activation conditions were used in Alternative I to avoid false positives caused by missing information. A reusable support module was developed for ACC-specific computational and reasoning support.

The proposed IRep and CRes schema was tested in representing and reasoning about quantitative regulatory requirements in Chapter 19 of IBC 2009 and design information in a two-story duplex apartment test case. Two experiments were conducted to test the schema using perfect information and imperfect information. Using perfect information, on the testing data, both Alternative I and Alternative II achieved 100% recall, precision, and F1-measure in noncompliance detection. It took less than one second to automatically check the 1,442 design facts in the BIM test case with the quantitative regulatory requirements in Chapter 19 of IBC 2009. Using imperfect information, on the testing data, Alternative I and Alternative II achieved 87.6%, 98.7%, and 92.8%, and 98.4%, 77.2%, and 86.5% precision, recall, and F1-measure, respectively. Alternative I blocks some false negatives and thus



results in a higher recall, while Alternative II blocks some false positives and thus results in a higher precision. Because high recall is more important than high precision in ACC, to avoid missing noncompliance instances, Alternative I is more suitable for ACC applications. The final proposed IRep and CRes schema (following Alternative I), thus, achieved 87.6% and 98.7% precision and recall, respectively, on the testing data, for noncompliance detection. This shows that the proposed IRep and CRes schema combined with the proposed automated regulatory information extraction and regulatory information transformation algorithms could achieve high recall and precision in noncompliance detection. An error analysis also pinpointed the sources of errors in the experimental results.

#### **9.1.6 Conclusions for the Prototype System**

All developed methods and algorithms were successfully integrated and implemented in one platform (Java platform), forming a proof-of-concept prototype system that conducts ACC based on building code text (.txt file) and a BIM design model (.ifc file): the Semantic Natural Language Processing-based Automated Compliance Checking (SNACC) system.

To evaluate the SNACC system, a test case was used which included (1) a randomly selected Chapter 19 of IBC 2009, (2) a BIM (i.e., .ifc file) containing design information to be checked for compliance with the regulatory provisions in the selected chapter (see Section 6.3.1 for used test case), and (3) a compliance report gold standard based on a manual comparison of the design information with the regulatory provisions in the selected chapter.

The experimental results showed that the automatically-generated compliance report by the prototype system achieved 87.6% precision, 98.7% recall, and 92.8% F1-measure in noncompliance detection. This high recall in noncompliance detection shows that the SNACC system is promising. An error analysis identified that all errors were attributed to errors in the regulatory information extraction and regulatory information transformation modules, no errors were attributed to the design information extraction and transformation module or the compliance reasoning module.

## **9.2 Contributions to the Body of Knowledge**

### **9.2.1 Contributions of the Proposed Regulatory Information Extraction Method and Algorithm**

This research contributes to the body of knowledge in four main ways. First, this research offers a domain-specific, semantic NLP method that can assist in capturing domain-specific meaning and shows that ontology-based semantic IE outperforms syntactic-only IE (in terms of precision and recall). Domain-specific semantics allow for the analysis of complex sentences that would otherwise be too complex for automated IE, the recognition of domain-specific text meaning, and in turn the improvement of IE performance. Second, this research offers relatively efficient-to-develop rule-based NLP methods that can benefit from expert NLP knowledge encoded in the form of IE and CR rules. This research shows that the efficiency of algorithm development for rule-based methods can be enhanced through the

following two main techniques: (1) use of PSG-based phrasal tags, and (2) separation and sequencing of semantic information elements (SSSIE) during extraction. Both PSG-based phrasal tags and the SSSIE method reduce the number of patterns needed in IE rules, resulting in fewer IE rules for extraction being required and, thus, reduced human effort to develop IE rules. Third, this research shows that deep NLP can be successfully achieved if both domain knowledge (represented in the form of a domain ontology) and expert NLP knowledge (represented in the form of IE and CR rules) are captured and integrated in a single platform. The research shows that semantic, rule-based deep NLP can provide high IE performance results (96.9% and 94.4% precision and recall, respectively). Fourth, and most importantly, this study is the first in the AEC domain that addresses automated IE using a semantically deep NLP approach. It offers baseline semantic IE methods/algorithms for extracting information from textual construction documents. Future research could use these methods/algorithms as a benchmark and build on this work by adapting the developed algorithms to extract information from other types of documents (e.g., contract documents) or for different purposes (e.g., contract analysis). The IE rules, CR rules, and algorithms developed in this study are potentially reusable (as the experimental results showed). Compared with the author's initial efforts, future efforts in adapting the rules and/or algorithms should be significantly lower. Once the rules/algorithms are adapted (if needed), the process of information extraction is fully automated.

## **9.2.2 Contributions of the Proposed Regulatory Information Transformation Method and Algorithm**

This research contributes to the body of knowledge in four main ways. First, a domain-specific, semantic NLP-based information processing method that can achieve full-sentence processing as opposed to partial-sentence processing (i.e., only specific terms or concepts are processed) is offered. Domain-specific semantics allow for analyzing complex sentence structures that would otherwise be too complex and ambiguous for automated ITr, recognizing domain-specific text meaning, and in turn allowing for processing and understandability of full sentences. Full sentence processing and understandability allows for a deeper level of NLP, namely, natural language understanding. Second, this research shows that a hybrid approach that combines rule-based NLP methods and semantic NLP methods could achieve high performance for the combination of IE and ITr from/of regulatory text in spite of the complexity inherent in natural language text. Domain-specific expert NLP knowledge (encoded in the form of rules) along with domain knowledge (represented in the form of an ontology) facilitates deep text processing and understandability. This research shows high performance for rule-based, semantic ITr. Third, a new context-aware and flexible way of utilizing pattern-matching-rule based methods is offered. This way of utilizing pattern-matching based rules captures the details (in terms of the expression and language structure) of complex sentence components through the use of context-aware semantic mapping rules and flexible pattern lengths. Fourth, a new mechanism (consume and

generate mechanism) for processing and transforming complex regulatory text into logic clauses is offered. The proposed mechanism follows the bottom-up method, which has shown based on the experimental results to outperform the top-down method in ITr. The high performance that the mechanism achieved verifies that the bottom-up method is suitable for such ITr tasks.

### **9.2.3 Contributions of the Proposed IFC Extension Method and Algorithm**

This research contributes to the body of knowledge in four main ways. First, this research offers a method for automated concept extraction that utilizes POS-pattern-matching-based rules to extract regulatory concepts from natural language regulatory documents. The set of POS patterns that was developed captures natural language knowledge, which allows for the recognition of concepts based on the lexical and functional categories of their terms. The pattern set includes only flattened patterns to avoid recursive parsing, which allows for efficient computation. The set of POS patterns are also generalized, and thus can be used to extract concepts in other domains. Second, this research offers a matching-based method for identifying and selecting the most related IFC concepts to the extracted regulatory concepts. The proposed method leverages both syntactic and semantic knowledge, which allows for the recognition of related concept pairs based on the syntactic and semantic similarities of their terms. As part of this method, two new concept-level semantic similarity (SS) scoring functions are offered. In the context of schema extension, existing SS scoring functions allow for measuring SS at the term-level. These proposed two functions further allow for measuring

SS at the concept-level. Third, this research offers an automated machine learning classification method for classifying the relationships between the extracted regulatory concepts and their most related IFC concepts. The classification results show that semantic features could benefit the task of relationship classification and result in further improvement of precision. The proposed method is also generalized and can be used to classify the relationships between any two concepts, based on eight syntactic and semantic features of their terms, into the following four types: equivalent concept, superconcept, subconcept, and associated concept. Fourth, the experimental results show that the three proposed methods could be effectively combined in a sequential way for extending the IFC schema with regulatory concepts from regulatory documents. This offers a new method for objectively extending the IFC schema with domain-specific concepts that are extracted from natural language documents. The proposed combined method is also generalized and can be used to extend the IFC schema with other types of concepts (e.g., environmental concepts) from other types of documents (e.g., environmental documents) or to extend other types of class hierarchies (e.g., of an ontology) in the construction domain or in other domains.

#### **9.2.4 Contributions of the Proposed BIM Information Extraction and Transformation**

##### **Method and Algorithm**

This research contributes to the body of knowledge by offering a BIM information extraction and transformation method that enables direct flow of design information from .ifc files to logic representations. As a result, this method allows for direct extraction of IFC-represented

data into logic facts. This enables information transfer between BIMs and logic programs. In addition to supporting ACC, the combined capabilities of building information modeling and logic programming could allow for the use of BIM information in an intelligent way and could open the door to more utilization of building information modeling in various automated applications in the construction domain such as automated cost analysis, schedule analysis, and facility maintenance decision analysis.

### **9.2.5 Contributions of the Proposed Information Representation and Compliance Reasoning Schema**

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

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

#### **9.2.6 Contributions of the Prototype System**

This research contributes to the body of knowledge by showing that all developed methods and algorithms can be successfully integrated and implemented in a proof-of-concept prototype system: the Semantic Natural Language Processing-based Automated Compliance Checking (SNACC) system. This work empirically shows that when tested using Chapter 19 of IBC 2009 and a BIM test case (see Section 6.3.1 for the test case that was used), the prototype system achieved 87.6% precision, 98.7% recall, and 92.8% F1-measure in noncompliance detection, which shows that the system is promising.



### **9.3 Limitations and Recommendations for Future Research**

#### **9.3.1 Limitations of the Proposed Regulatory Information Extraction Method and Algorithm and Recommendations for Future Research**

Three limitations of the work are acknowledged. First, in compliance with the scope of the dissertation, the proposed method/algorithm was only tested in extracting quantitative requirements. The types of patterns and extraction conflicts in other types of requirements (e.g., existential requirements) may vary and, as a result, information extraction performance may vary. In future research, the method/algorithm could be tested on other types of requirements such as existential requirements. Second, the proposed method/algorithm was only tested on one chapter, primarily because the development of the gold standard for testing is highly time intensive. When the method/algorithm is tested on more building codes chapters, the results are expected to show similar high performance because the chapter used in testing contains large amounts of text (approximately 7,000 words) and because of the similarity in text across different chapters of building codes and across different types of building codes (e.g., “Building Code and Related Excerpts of the Municipal Code of Chicago” versus IBC 2006). However, the results may vary because of the possible variability in the syntactic and semantic text features across different chapters and/or codes. In that case, the proposed IE and CR rules may be adapted/extended based on additional development text. Third, in compliance with the scope of the dissertation, the proposed method/algorithm was tested only on building codes. In future research, the proposed method/algorithm could be

extended to extract information from other types of regulatory documents (e.g., environmental regulations) and contractual documents (e.g., contract specifications).

### **9.3.2 Limitations of the Proposed Regulatory Information Transformation Method and Algorithm and Recommendations for Future Research**

Three main limitations of this work are acknowledged. First, in compliance with the scope of the dissertation, the methodology was only tested on processing quantitative requirements. The types of semantic patterns and conflicts in other types of requirements (e.g., existential requirements) may vary and, thus, may lead to different performance results. Although the processing of other types of requirements is expected to be less or equally complex than that of quantitative requirements – and thus is expected to have similar or better performance, the proposed information transformation method needs to be tested on other types of requirements (e.g., existential requirements) for validation. Second, due to the large amount of manual effort required in developing a gold standard, the proposed information transformation algorithm was tested only on one chapter of IBC 2009. Similar high performance is expected when testing on other chapters of IBC and on other regulatory documents, since all regulatory documents share similarities in expressions. However, different performance results might be obtained due to the possible variability of text across different chapters or different regulatory documents. As such, future research is needed to test the proposed information transformation method/algorithm on more chapters of IBC 2009. Third, in compliance with the scope of the dissertation, the proposed method/algorithm was

tested only on building codes. Future research is needed to test the proposed method/algorithm on other types of regulatory documents (e.g., environmental regulations) and contractual documents (e.g., contract specifications).

### **9.3.3 Limitations of the Proposed IFC Extension Method and Algorithm and Recommendations for Future Research**

Two limitations of the proposed semiautomated IFC extension method are acknowledged. First, due to the large amount of manual effort required in developing the gold standard for each phase, the proposed method was only tested on one Chapter of IBC 2009. Similar high performance is expected on other chapters of IBC and other regulatory documents. However, different performance results might be obtained due to the possible variability of contents across different chapters of IBC 2009 or across different types of regulatory documents. As such, in future research, the proposed method needs to be tested on more chapters of IBC 2009 and on other types of regulatory documents (e.g., EPA regulations). Second, only unigram (single terms) semantic-based matching was used for finding semantically related F-concepts to an R-concept. While the combinatorial nature of term meanings [i.e., the meanings of single terms (e.g., “exterior” and “door”) in a concept name are combined to form the overall meaning of the whole concept (e.g., “exterior door”)] renders this unigram method effective, there may be cases where bigram (pairs of terms) or multigram (groups of three or more terms) matching could be effective. As such, in future research, the semantic-based matching method needs to be extended to incorporate semantic relations

between bigrams and multigrams to test whether such bigram or multigram considerations could further improve the performance of concept matching.

### **9.3.4 Limitations of the Proposed BIM Information Extraction and Transformation**

#### **Method and Algorithm and Recommendations for Future Research**

Two main limitations of the proposed BIM information extraction and transformation method are acknowledged. First, due to the large amount of manual effort required in developing the gold standard for information extraction and transformation, the proposed method was tested only on one BIM test case. Although similar high performance is expected on other test cases, further testing and evaluation of the proposed method on more BIM test cases is recommended for verification. Second, extracting information from BIM models needs further exploration to find out how the different BIM implementation platforms (e.g., Autodesk Revit, ARCHIBUS EIM with BIM 4.0, ArchiCAD) and the different levels of information completeness in the instances of those BIM models (i.e., what information was entered or not entered into the BIM model/platform) may affect the need for extra model/information processing (e.g., model extension). As such, future research is recommended to test the proposed method on more BIM test cases, different types of BIM implementation platforms, and different levels of information completeness.

### **9.3.5 Limitations of the Proposed Information Representation and Compliance Reasoning Schema and Recommendations for Future Research**

Two main limitations of this work are acknowledged. First, in compliance with the scope of the dissertation, the work focused on quantitative regulatory requirements. While the literature suggests that qualitative regulatory requirements could be transformed into quantitative requirements, the transformation details and its implication on the representation and reasoning methodology needs further exploration. Second, due to the large amount of manual effort needed in compiling test cases and developing a gold standard, the proposed schema was only tested on one chapter of IBC 2009 and one BIM test case. While similar performance could be expected on other chapters of IBC 2009, on other regulatory documents, and other BIM test cases, more empirical tests on other chapters of IBC 2009, other regulatory documents, and other BIM test cases are needed for verification, especially when using imperfect information. Future research is recommended to address these limitations by: (1) implementing/adapting the proposed schema on qualitative requirements and developing corresponding information extraction and information transformation algorithms; and (2) testing the proposed schema on more chapters of IBC 2009, other construction regulatory documents, and other BIM test cases.

### **9.3.6 Limitations of the Prototype System and Recommendations for Future Research**

Two main limitations of this system are acknowledged. First, the SNACC system needs the

design information in the input BIM to be sufficient for compliance checking. How insufficient design information are to be addressed for ACC still needs investigation, both theoretically and empirically. Second, due to the large amount of effort needed in developing gold standards, the SNACC system was tested only on one test case. How different types of regulatory and design information would affect the performance of the SNACC system needs further testing. Future research is recommended to address these limitations by: (1) investigating the feasible and/or optimal solutions to address insufficient design information in BIM for ACC, both theoretically and empirically; and (2) developing more gold standards and test cases using different types of regulatory and design information to further test the SNACC system.

## 10 REFERENCES

- Abney, S. (1997). "Part-of-speech tagging and partial parsing." *Text Speech Lang. Technol.*, 2, 118-136.
- Abuzir, Y., and Abuzir, M.O. (2002). "Constructing the civil engineering thesaurus (CET) using ThesWB." *Proc., Int. Workshop Information Tech. Civ. Eng. 2002*, ASCE, Reston, VA, 400-412.
- Afrin, T. (2001). "Extraction of basic noun phrases from natural language using statistical context-free grammar." Master of Science Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.
- AISC, (2014). "Technology integration." <<http://www.aisc.org/content.aspx?id=26044>>. (Jun. 17, 2014).
- Akinci, B., Fischer, M., and Kunz, J. (2002). "Automated generation of work spaces required by construction activities." *J. Constr. Eng. and Manage.*, 128(4), 306-315.
- Al Qady, M.A., and Kandil, A. (2010). "Concept relation extraction from construction documents using natural language processing." *J. Constr. Eng. Manage.*, 136(3), 294-302.
- Al-Yahya, M. (2014). "Ontology-based multiple choice question generation." *The Scientific World Journal*, 2014, 274949.
- Alexander, S.B., and Sidney, H.S. (1987). "KBES for the design of reinforced concrete columns." *Structural Eng. Report*, 152(1987), 149-164.
- Amann, J., Singer, D., and Borrmann, A. (2015). "Extension of the upcoming IFC alignment standard with cross sections for road design." <[http://www.cms.bgu.tum.de/publications/2015\\_Amann\\_CrossSections.pdf](http://www.cms.bgu.tum.de/publications/2015_Amann_CrossSections.pdf)> (Nov. 20, 2015).
- Anjomshoaa, A., Shayeganfar, F., Mahdavi, A., and Tjoa, A.M. (2015). "Towards constructive evidence of linked open data in AEC domain." *eWork and eBusiness in Architecture, Engineering and Construction – Martens, Mahdavi & Scherer (Eds)*, Taylor & Francis Group, London, 535-541.
- Anumba, C.J., Issa, R.R.A., Pan, J., and Mutis, I. (2008). "Ontology-based information and knowledge management in construction." *Constr. Innovation*, 8(3), 218-239.
- Avolve Software Corporation. (2011). *Electronic plan review for building and planning departments*.

- <<http://www.avolvesoftware.com/index.php/solutions/building-departments/>> (Jul. 15, 2012).
- Awad, A., Smirnov, S., and Weske, M. (2009). "Resolution of compliance violation in business process models: a planning-based approach." *Proc., OTM '09*, 6-23.
- Banko, M., Brill, E., Dumais, S., and Lin, J. (2002). "AskMSR: question answering using the worldwide web." *Proc., 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, Palo Alto, California, 7-9.
- Batres, R., Fujihara, S., Shimada, Y., and Fuchino, T. (2014). "The use of ontologies for enhancing the use of accident information." *Process Safety and Environmental Protection*, 92(2014), 119-130.
- Baumgartner, P., and Suchanek, F.M. (2006). "Automated reasoning support for first-order ontologies." *Proc., 4th Int. Conf. on Principles and Practice of Semantic Web Reasoning*, Springer-Verlag Berlin, Heidelberg, Germany, 18-32.
- Beach, T.H., Kasim, T., Li, H., Nisbet, N., and Rezgui, Y. (2013). "Towards automated compliance checking in the construction industry." *H. Decker et al. (Eds.): DEXA 2013, Part I, LNCS 8055*, 366-380.
- Beetz, J., Leeuwen, J.V., and Vries, B.D. (2009). "IfcOWL: A case of transforming EXPRESS schemas into ontologies." *Artificial Intelligence for Eng. Design, Analysis and Manufacturing (2009)*, 23, 89-101.
- Bell, H., Bjorkhaug, L., and Hjelseth. (2009). "Standardized Computable Rules." Oslo, Norway.
- Bernier-Colborne, G. (2012). "Defining a gold standard for the evaluation of term extractors." *Proc., the ColabTKR Workshop, LREC*, European Language Resources Association (ELRA), Paris, France.
- BIMForum. (2013). "Level of development specification." <[www.bimforum.org/lod](http://www.bimforum.org/lod)> (Jun. 28, 2015).
- Bing, L., Lam, W., and Wong, T. (2013). "Wikipedia entity expansion and attribute extraction from the web using semi-supervised learning." *Proc., 6th ACM Int. Conf. Web Search and Data Mining (WSDM '13)*, ACM, New York, NY, 567-576.
- Bishop, C.M. (2006). "Pattern recognition and machine learning." Springer Science+Business Media, LLC, New York, NY.



- Bird, S., Klein, E., and Loper, E. (2009). "Natural language processing with Python." O'Reilly Media Inc., Sebastopol, CA.
- Boken, P., and Callaghan, G. (2009). "Confronting the challenges of manual journal entries." Protiviti, Alexandria, VA, 1-4.
- Bos, J. (2009). "Applying automated deduction to natural language understanding." *J. Applied Logic*, 7(2009), 100-112.
- Boukamp, F., and Akinci, B. (2007). "Automated processing of construction specifications to support inspection and quality control." *Autom. Constr.*, 17(1), 90-106.
- Brank, J., Grobelnik, M., and Mladenic, D. (2005). "A survey of ontology evaluation techniques." *SiKDD 2005*, Information Society, Ljubljana, Slovenia.
- Breaux, T. D., and Anton, A. I. (2008). "Analyzing regulatory rules for privacy and security requirements." *IEEE Trans. Software Eng.*, 34(1), 5-20.
- Bui, D.D.A., and Zeng-Treitler, Q. (2014). "Learning regular expressions for clinical text classification." *J. Am. Med. Inform. Assoc.*, 2014(21), 850-857.
- BuildingSmart, (2014). "Industry Foundation Classes (IFC) data model." <<http://www.buildingsmart.org/standards/ifc/model-industry-foundation-classes-ifc>> (Jun. 17, 2014).
- BuildingSmart, (2015). "IFC overview summary." <http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-overview-summary> (Nov. 22, 2015).
- Buitelaar, P., Cimiano, P., Frank, A., Hartung, M., and Racioppa, S. (2008). "Ontology-based information extraction and integration from heterogeneous data sources." *Int. J. Human-Comput. Studies*, 66(11), 759-788.
- Bundy, A. (2013). "The interaction of representation and reasoning." *Proc., R. Soc. A*, 469(2157), 1-18.
- Cai, Q., and Yates, A. (2013). "Large scale semantic parsing via Schema Matching and Lexicon Extension." *Proc., the Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Stroudsburg, PA.
- Caldas, C.H., and Soibelman, L. (2003). "Automating hierarchical document classification for construction management information systems." *Autom. Constr.*, 12(2003), 395-406.
- Califf, M. E., and Mooney, R. J. (2003). "Bottom-up relational learning of pattern matching

- rules for information extraction.” *J. Mach. Learn. Res.*, 4, 177-210.
- Carletta, J. (1996). “Assessing agreement on classification tasks: the Kappa statistic.” *Journal Computational Linguistics*, 22(2), 1996, 249-254.
- Cheng, J.C.P., and Das, M. (2013). “A cloud computing approach to partial exchange of BIM models.” *Proc., CIB W78 2013*, Conseil International du Bâtiment (CIB), Rotterdam, Netherlands.
- Cherpas, C. (1992). “Natural language processing, pragmatics, and verbal behavior.” *Analysis of Verbal Behavior*, 10,135-147.
- Chevallier, N. and Russell, A.D. (1998). “Automated schedule generation.” *Canadian J. Civ. Eng.*, 25(6), 1059-1077.
- Chieu, H.L., and Ng, H.T. (2003). “Named entity recognition with a maximum entropy approach.” *Proc., CoNLL-2003*, Association for Computing Machinery, New York.
- Chomsky, N. (1956). “Three models for the description of language.” *Information Theory, IEEE Transactions*, 2(3), 113-124.
- City of Champaign, (2015). “Building codes and amendments.” <<http://ci.champaign.il.us/departments/fire/building-safety/building-codes-and-amendments/>> (Feb. 4, 2015).
- City of Mesa. (2012). “Construction plan review.” *The Official Website of the City of Mesa, Arizona*. <<http://www.mesaaz.gov/devsustain/PlanReview.aspx>> (Nov. 25, 2012).
- Clarke, J., Goldwasser, D., Change, M., and Roth, D. (2010). “Driving semantic parsing from the world’s response.” *CoNLL’10*, Association for Computing Machinery, New York.
- Corcho, O., Fernandez, M., and Gomez-Perez, A. (2003). “Methodologies, tools and languages for building ontologies: where is the meeting point?” *Data & Knowledge Engineering*, 46, 2003.
- Corke, G. (2013). “Solibri Model Checker V8”. *AECMagazine, Building Information Modelling (BIM) for Architecture, Engineering and Construction*. <<http://aecmag.com/index.php?option=content&task=view&id=527>> (May 19, 2013).
- Costa, V.S. (2009). “On just in time indexing of dynamic predicates in Prolog.” *Prog. AI, Lect. Notes. Comput. Sc.*, 5816(2009), 126-137.
- Costa, G., and Madrazo, L. (2015). “An information system architecture to create building components catalogues using semantic technologies.” *eWork and eBusiness in*

- Architecture, Engineering and Construction – Martens, Mahdavi & Scherer (Eds)*, Taylor & Francis Group, London, 551-557.
- Cover, T.M. (1967). “Nearest neighbor pattern classification.” *IEEE Transactions on Information Theory*, 13(1), 21-27.
- Covington, M.A. (2001). “A fundamental algorithm for dependency parsing.” *Proc., 39th Annual ACM Southeast Conf.*, ACM, Athens, Georgia, 95-102.
- Cristianini, N., and Shawe-Taylor, J. (2000). “An introduction to support vector machines and other kernel-based learning methods.” Cambridge University Press, 1st edition, Cambridge, U.K.
- Cunningham, H. et al. (2011). “Developing language processing components with gate version 6 (a user guide).” The University of Sheffield, Department of Computer Science, Sheffield, U.K.
- Cunningham, H. et al. (2012). “Developing language processing components with gate version 7 (a user guide).” The University of Sheffield, Department of Computer Science, Sheffield, U.K.
- Deleger, L., Lingren, T., Ni, Y., Kaiser, M., Stoutenborough, L., Marsolo, K., Kouril, M., Molnar, K., and Solti, I. (2014). “Preparing an annotated gold standard corpus to share with extramural investigators for de-identification research.” *Journal of Biomedical Informatics*, 50 (2014), 173-183.
- Deleger, L., Molnar, K., Savova, G., Xia, F., Lingren, T., Li, Q., Marsolo, K., Jegga, A., Kaiser, M., Stoutenborough, L., and Solti, I. (2013). “Large-scale evaluation of automated clinical note de-identification and its impact on information extraction.” *J. Am. Med. Inform. Assoc.*, 20(1), 84-94.
- Delgado, F., Martínez-González, M.M., and Finat, J. (2013). “An evaluation of ontology matching techniques on geospatial ontologies.” *Int. J. Geogr. Inf. Sci.*, 27(12), 2279-2301.
- Delis, E.A., and Delis, A. (1995) “Automatic fire-code checking using expert-system technology.” *J. Comput. Civ. Eng.*, 9(2), 141-156.
- Denecker, M., Vennekens, J., Vlaeminck, H., Wittocx, J., and Bruynooghe, M. (2011). “Answer set programming’s contributions to classical logic: an analysis of ASP methodology.” *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, Springer-Verlag, Berlin, Heidelberg, 12-32.
- Deokar, A.V., and Sen, S. (2010). “Ontology-based information extraction for analyzing IT

- services.” *Proc. ICIS 2010*, Paper 216.
- DeYoung, H., Garg, D., Kaynar, D., and Datta, A. (2010). “Logical specification of the GLBA and HIPAA privacy laws.” Carnegie Mellon University, Pittsburgh, PA.
- Dimyadi, J., and Amor, R. (2013). “Automated building code compliance checking - where is it at?” *Proc. 19th Int. CIB World Build. Congress*, Brisbane, Australia.
- Dimyadi, J., Clifton, C., Spearpoint, M., and Amor, R. (2014). “Regulatory knowledge encoding guidelines for automated compliance audit of building engineering design.” *Comput. Civ. Build. Eng. (2014)*, ASCE, Reston, VA, 536-543.
- Ding, L., Drogemuller, R., Rosenman, M., Marchant, D., and Gero, J. (2006). “Automating code checking for building designs – DesignCheck.” *Clients Driving Innovation: Moving Ideas into Practice*, CRC for Construction Innovation, Brisbane, Australia, 1-16.
- Domingos, P. (2012). “A few useful things to know about machine learning.” *Communications of the ACM*, 55(10), 78-87.
- Dzeng, R.J., Tserng, H.P., and Wang, W.C. (2005). “Automating schedule review for expressway construction.” *J. Constr. Eng. Manage.*, 131(1), 127-136.
- East, E.W. (2013). “Common building information model files and tools.” <[http://www.nibs.org/?page=bsa\\_commonbimfiles&hhSearchTerms=%22common+and+BIM+and+file%22](http://www.nibs.org/?page=bsa_commonbimfiles&hhSearchTerms=%22common+and+BIM+and+file%22)> (Jun. 27, 2014).
- Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009) “Automatic rule-based checking of building designs.” *Autom. Constr.*, 18(8), 1011-1033.
- El-Diraby, T.E., and Kashif, K.F. (2005). “Distributed ontology architecture for knowledge management in highway construction.” *J. Constr. Eng. Manage.*, 131(5), 591-603.
- El-Diraby, T.E., and Zhang, J. (2006). “A semantic framework to support corporate memory management in building construction.” *Autom. Constr.*, 15(4), 504-521.
- El-Gohary, N.M., and El-Diraby, T.E. (2010). “Domain ontology for processes in infrastructure and construction.” *J. Constr. Eng. Manage.*, 136(7), 730-744.
- Eurostep. (2002). “IFCToolboX: Extended tool support for IFC implementations.” <[http://cic.vtt.fi/vera/projects/e\\_ifctoolbox.htm](http://cic.vtt.fi/vera/projects/e_ifctoolbox.htm)> (Jun. 15, 2015).
- Facilities Information Council (FIC). (2007). “Automated code compliance checking (AC3).” A National Building Information Model Standard Project Fact Sheet, National Institute of Building Sciences, Washington, DC.

- Farkas, R., Vincze, V., Mora, G., Csirik, J., and Szarvas, G. (2010). "The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text." *Proc., the Fourteenth Conference on Computational Natural Language Learning: Shared Task*, Association for Computational Linguistics, Stroudsburg, PA, 1-12.
- Fautsch, C., and Savoy, J. (2009). "Algorithmic stemmers or morphological analysis?" *J. Am. Soc. Inform. Sci. and Tech.*, 60(8), 1616-1624.
- Fellbaum, C. (2005). "WordNet and wordnets." In: Brown, Keith et al. (eds.), *Encyclopedia of Language and Linguistics*, Second Edition, Oxford: Elsevier, 665-670.
- Fenves, S.J., Gaylord, E.H., and Goel, S.K. (1969). "Decision table formulation of the 1969 AISC specification." *Civ. Eng. Studies: Structural Research Series*, 347.
- Fernandez, M., Gomez-Perez, A. and Juristo, N. (1997). "METHONTOLOGY: from ontological art towards ontological engineering." *AAAI-97 Spring Symposium on Ontological Engineering*, AAAI, Palo Alto, California, 33-40.
- Fiatech. (2015). "Fiatech current projects - summary descriptions." <<http://www.fiatech.org/projects/active-projects>> (Feb. 6, 2015).
- Fiatech. (2014). "AutoCodes - release of first iteration of model matrix." <<http://www.fiatech.org/impact-alert-project-management>> (Feb. 6, 2015).
- Fiatech. (2012). "AutoCodes project: phase 1, proof-of-Concept final report." <[http://www.fiatech.org/images/stories/techprojects/project\\_deliverables/Updated\\_project\\_deliverables/AutoCodesPOCFINALREPORT.pdf](http://www.fiatech.org/images/stories/techprojects/project_deliverables/Updated_project_deliverables/AutoCodesPOCFINALREPORT.pdf)> (Feb. 6, 2015).
- Fiatech. (2011). "Fiatech AutoCodes project phase ii – participation and funding." <<http://www.fiatech.org/images/stories/projects/Autocodesfundingprospectus.pdf>> (Feb. 6, 2015)
- Freund, Y., and Schapire, R. (1999). "Large margin classification using the perceptron algorithm." *Mach. Learn.*, 37(3), 277-296.
- Fritz, D. (2006). "The Semantic Model: A basis for understanding and implementing data warehouse requirements." <<http://www.tdan.com/view-articles/4044>> (Aug. 12, 2014).
- Galasso, J. (2002). "Analyzing English grammar: an introduction to feature theory, a companion handbook." California State University Northridge, Northridge, CA.
- Gallaher, M.P., O'Connor, A.C., Dettbarn, J.L., Jr., and Gilday, L.T. (2004). "Cost analysis of inadequate interoperability in the U.S. capital facilities industry." NIST GCR 04-867,

NIST, Gaithersburg, MD.

- Garg, D., Jia, L., and Datta, A. (2011). "Policy auditing over incomplete logs: theory, implementation and applications." *Proc., 18th ACM Conf. Computer and Communications Security*, ACM, New York, NY, 151-162.
- Garrett, Jr., J.H., and Fenves, S.J. (1987). "A knowledge-based standard processor for structural component design." *Eng. Comput.*, 2(4), 219-238.
- Garrett, J.H.Jr., and Palmer, M.E. (2014). "Delivering the Infrastructure for Digital Building Regulations." *J. Comput. Civ. Eng.*, 2014(28), 167-169.
- Goh, O. S., Depickere, A., Fung, C. C., and Wong, K. W. (2006). "Topdown natural language query approach for embodied conversational agent." *Proc., Int. Multiconference Engineering and Computer Science (IMECS 2006)*, International Association of Engineers (IAENG), Hong Kong.
- Gordon, C., Akinci, B., and Garrett, Jr., J.H. (2008). "Automated planning support for on-site construction inspection." *Autom. Constr.*, 17(2008), 705-718.
- Goutte, C., and Gaussier, E. (2005). "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation." *Proc., 27th European conference on IR research (ECIR 2005)*, Springer-Verlag Berlin Heidelberg, Germany, 345-359.
- Grefenstette, G., and Tapanainen, P. (1994) "What is a word, what is a sentence? problems of tokenization." *Proc., 3rd Conf. Comput. Lexicography and Text Research (COMPLEX'94)*, Research Institute for Linguistics Hungarian Academy of Sciences, Budapest, Hungary, 79-87.
- Gruber, T.R. (1995). "Toward principles for the design of ontologies used for knowledge sharing." *Int. J. Human-Comput. Studies*, 43, 907-928.
- Gunavathi, C., and Premalatha, K. (2014). "Performance analysis of genetic algorithm with kNN and SVM for feature selection in tumor classification." *International Journal of Computer, Control, Quantum and Information Engineering*, 8(8), 1397-1404.
- Grunewald, J., Kaiser, J., and Hoch, R. (2010). "HESMOS Deliverable D5.3: Full prototype of the energy simulation tools with interface to IVEL and CAD." *Seventh Framework Programme of the European Community*, HESMOS Consortium, Brussels, Belgium.
- Gruninger, M., and Fox, M.S., (1995). "Methodology for the design and evaluation of ontologies." *Proc., Workshop Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal, Canada.

- Guo, Y., Li, Y., and Shao, Z. (2012). "Cognitive intentionality extraction from discourse with pragmatic-tree construction and analysis." *Information Sciences*, 214 (2012), 35-55.
- Gupta, R., and Kochenderfer, M.J. (2004). "Common sense data acquisition for indoor mobile robots." *Nineteenth National Conf., Artificial Intelligence, AAAI Press*, Palo Alto, California, 605-610.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. (2009). "The WEKA data mining software: an update." *SIGKDD Explor.*, 11(1), 10-18.
- Halpern, J.Y., and Weissman, V. (2008). "Using first-order logic to reason about policies." *J. ACM Trans. Inf. Sys. Security (TISSEC)*, 11(4), 1-41.
- Hamil, S. (2012). "Building information modelling and interoperability." <<http://www.thenbs.com/topics/bim/articles/bimAndInteroperability.asp>> (Aug. 12, 2014).
- Han, C.S., Kunz, J.C., and Law, K.H. (1997). "Making automated building code checking a reality." *J. Manage.*, September/October, 22-28.
- Hanis, T., and Noller, D. (2011). "The role of semantic models in smarter industrial operations." *Oper.*, IBM Corporation, Armonk, New York.
- Hauptmann, A.G., Jin, R., and Ng, T.D. (2003). "Video retrieval using speech and image information." *Storage and Retrieval for Media Databases (Proc. SPIE/IS&T Volume 5021)*, Santa Clara, CA, 148-159.
- Hjelseth, E. and Nisbet, N. (2011). "Capturing normative constraints by use of the semantic mark-up RASE methodology." *Proc., CIB W78 2011*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.
- Hodges, W. (2001). "Classical logic I - first-order logic." *Goble*, 2001, 9-32. <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.4783&rep=rep1&type=pdf>> (Dec. 26, 2013).
- Hofmann, M. (2006). "Support vector machines - kernels and the kernel trick." <[http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs\\_svm/slides/SVM\\_Seminarbericht\\_Hofmann.pdf](http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf)> (Jun. 24, 2015).
- Hogenboom, A., Hogenboom, F., Frasinca, F., Schouten, K., and Meer, O.V.D. (2013). "Semantics-based information extraction for detecting economic events." *Multimed Tools*, 2013(64), 27-52.
- Horton, J.D., and Spencer, B. (1997). "Clause trees: a tool for understanding and

- implementing resolution in automated reasoning.” *Artif. Intel.*, 92, 8-9.
- Huang, Y.J., Powers, R., and Montelione, G.T. (2005). “Protein NMR recall, precision, and F-measure scores (RPF scores): structure quality assessment measures based on information retrieval statistics.” *J. AM. CHEM. SOC.*, 127, 1665-1674.
- Humphreys, K., Demetriou, G., and Gaizauskas, R. (2000). “Two applications of information extraction to biological science journal articles: enzyme interactions and protein structures.” *Pacific Symposium on Biocomput.*, 5, 505-516.
- International Alliance for Interoperability (IAI). (2007). “IFC2x edition 3 technical corrigendum 1.” *Industry Foundation Classes*, <<http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm>> (July 15, 2011).
- International Code Council (ICC). (2012). “ICC Code Adoption Toolkit.” <<http://www.iccsafe.org/gr/Documents/AdoptionToolkit/toolkit.pdf>> (Feb. 4, 2015).
- International Code Council (ICC). (2011). “SMARTcodes.” <[http://web.stanford.edu/group/narratives/classes/08-09/CEE215/ReferenceLibrary/International%20Code%20Council%20\(ICC\)/SMARTcodes%20fact%20sheet%2011-01-07.pdf](http://web.stanford.edu/group/narratives/classes/08-09/CEE215/ReferenceLibrary/International%20Code%20Council%20(ICC)/SMARTcodes%20fact%20sheet%2011-01-07.pdf)> (Feb. 4, 2015).
- International Code Council (ICC). (2009). “2009 International Building Code.” *2009 Int. Codes*, <<http://publicecodes.cyberregs.com/icod/ibc/2009/index.htm>> (Feb. 05, 2011).
- International Code Council (ICC). (2007). “Take a SMARTcodes test drive.” <<http://www.iccsafe.org/newsroom/News%20Releases/1011smartcodes.pdf>> (Feb. 4, 2015).
- International Code Council (ICC). (2006). “2006 International Building Code.” *2006 Int. Codes*, <<http://publicecodes.citation.com/icod/ibc/2006f2/index.htm>> (Feb. 05, 2011).
- IFC Tools Project. (2013). <<http://www.ifctoolsproject.com/>> (Jun. 15, 2015).
- ifcPlusPlus. (2015). <<http://www.ifcplusplus.com/index.php>> (Jun. 15, 2015).
- Isikdag, U., Aouad, G., Underwood, J, and Wu, S. (2007). “Building information models: a review on storage and exchange mechanisms.” *Proc., 24th W78 Conf. & 5th ITCEDU Workshop & 14th EG-ICE Workshop, Bringing ITC Knowledge to Work*, International Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands, 135-144.
- ISO. (2004). “ISO 10303-11:2004 - Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS



- language reference manual.” <  
[http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=38047](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047)> (Feb. 24, 2015).
- Ivanovic, M., and Budimac, Z. (2014). “An overview of ontologies and data resources in medical domains.” *Expert Systems with Applications*, 41(2014), 5185-5166.
- Jain, D., Krawinkler, H., and Law, K.H. (1989). “Knowledge representation with logic.” CIFE Technical Report Number 13, Stanford University, Stanford, CA.
- Java Standard Edition Development Kit 6u45. [Computer Software]. Redwood Shores, CA, Oracle.
- Jiang, Y., Wang, X, and Zheng, H. (2014). “A semantic similarity measure based on information distance for ontology alignment.” *Inf. Sci.*, 278(2014), 76-87.
- Jones, D.M., Bench-Caponand, T.J.M., and Visser, P.R.S. (1998). “Methodologies for ontology development.” *Proc., the IT and KNOWS Conference of the 15th FIP World Computer Congress*, 1998.
- Jones, K.S. (1997). “What is the role of NLP in text retrieval?” *Natural language information retrieval*, Ed. T. Strzalkowski, Dordrecht: Kluwer(1999),1-24.
- Joshi, A.K. (1991). “Natural language processing.” *Sci.*, 253, 1242-1249.
- JSDAI 4.1.505.v201112201320 [Computer software]. LKSoftWare GmbH, Kuenzell, Germany.
- JSDAI ExpressCompilerCore 4.1.11.v201112201318 [Computer software]. LKSoftWare GmbH, Kuenzell, Germany.
- Juszczyszyn, K., and Kołaczek, G. (2009). “Deontic logic reasoning for maintaining ontology consistency in agent networks.” *Proc., Third KES Int. Symposium on Agent and Multi-Agent Systems: Tech. and Applications (KES-AMSTA '09)*, Uppsala, Sweden, 252-262.
- Jython 2.2.1. [Computer software]. <<http://www.jython.org/>> (May 02, 2015).
- Kamps, J., Marx, M., Mokken, R.J., and Rijke, M. (2004). “Using WordNet to measure semantic orientations of adjectives.” *Proc., LREC-04*, European Language Resources Association (ELRA), Paris, France, 1115-1118.
- Kartam, N.A., Levitt, R.E., and Wilkins, D.E. (1991). “Extending artificial intelligence

- techniques for hierarchical planning.” *J. Comput. Civ. Eng.*, 5(4), 464-477.
- Kasim, T., Li, H., Rezgui, Y., and Beach, T. (2013). “Automated sustainability compliance checking process: proof of concept.” *Proc., 13th Int. Conf. Constr. App. Vir. Real.*, Teesside University, Tees Valley, UK, 11-21.
- Kataoka, M. (2008). “Automated generation of construction plans from primitive geometries.” *J. Constr. Eng. Manage.*, 134(8), 592-600.
- KBSI (1994) “The IDEF5 ontology description capture method overview.” KBSI Report, KBSI, College Station, TX.
- Kerrigan, S., and Law, K.H. (2003). “Logic-based regulation compliance-assistance.” *Proc., Ninth Int. Conf. Artificial Intelligence and Law (ICAIL 2003)*, ACM, New York, NY, 126-135.
- Khemlani, L. (2005). “CORENET e-PlanCheck: Singapore's automated code checking system.” *AECbytes “Building the Future” Article*, <<http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html>> (Oct. 26, 2013).
- Kim, B., Park, S., Kim, H., and Lee, S. (2010). “Automatic extraction of apparent semantic structure from text contents of a structural calculation document.” *J. Comput. Civ. Eng.*, 24(3), 313-324.
- Kim, H., Anderson, K., Lee, S., and Hildreth, J. (2013). “Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology.” *Autom. Constr.*, 35(2013), 285-295.
- Kim, S. and Hovy, E. (2004). “Determining the sentiment of opinions.” *Proc., 20th Int. Conf. Comput. Linguistics*, Association for Computational Linguistics, Stroudsburg, PA.
- Kiyavitskaya, N., Zeni, N., Breaux, T. D., Anton, A. I., Cordy, J. R., Mich, L., and Mylopoulos, J. (2008). “Automating the extraction of rights and obligations for regulatory compliance.” *Lect. Notes Comput. Sci.*, 5231, 154-168.
- Kołaczek, G., and Juszczyszyn, K. (2010). “Deontic logic-based framework for ontology alignment in agent communities.” *J. Universal Comput. Sci.*, 16(1), 178-197.
- Kovahi, R., and Provost, F. (1998). “Glossary of terms.” *Machine Learning*, 30, 271-274.
- Krotzsch, M., Rudolph, S., and Schmitt, P.H. (2015). “A closer look at the semantic relationship between Datalog and description logics.” *Semantic Web*, 6(2015), 63-79.

- Lau, G.T., and Law, K.H. (2004). "An information infrastructure for comparing accessibility regulations and related information from multiple sources." *Proc., 10th Int. Conf. on Comput. Civ. Build. Eng.*, Weimar, Germany, 1-11.
- Lee, G. (2009). "Concept-based method for extracting valid subsets from an EXPRESS schema." *J. Comput. Civ. Eng.*, 2009(23), 128-135.
- Lee, J., and Jeong, Y. (2012). "User-centric knowledge representations based on ontology for AEC design collaboration." *Computer-Aided Design*, 44(2012), 735-748.
- Lee, S., Kim, K., and Yu, J. (2014). "BIM and ontology-based approach for building cost estimation." *Autom. Constr.*, 2014, 96-105.
- Lee, S.H., and Kim, B.G. (2011). "IFC extension for road structures and digital modeling." *Procedia Eng.*, 14(2011), 1037-1042.
- Levine, R., and Meurers, W. (2006). "Head-driven phrase structure grammar linguistic approach, formal foundations, and computational realization." In *Encyclopedia of Language and Linguistics (2nd Ed.)*, K. Brown, Ed. Oxford: Elsevier., Oxford, UK.
- Liddy, E.D. (2003). "Natural language processing." In *Encyclopedia of Library and Information Science (2nd Ed.)*, Marcel Decker, Inc. NY.
- Ling, X., and Weld, D.S. (2012). "Fine-grained entity recognition." *Proc., 26th AAAI Conf. Artif. Intel.*, Association for the Advancement of Artificial Intelligence, Palo Alto, California, 94-100.
- Liu, H., and Singh, P. (2003). "OMCSNet: A commonsense inference toolkit." *MIT Media Lab Society Of Mind Group Technical Report SOM02-01*, Cambridge, MA, 272-277.
- Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., and Lee, B. (2012). "TwiNER: named entity recognition in targeted twitter stream." *Proc., 35th Int. ACM SIGIR Conf., Research and Development in Information Retrieval (SIGIR '12)*., ACM, New York, NY, 721-730.
- Li, T. (2010). "Practice and exploration of ontology creation algorithms." <  
[http://www.cs.ubc.ca/~carenini/TEACHING/CPSC503-14/FINAL-REPORTS-10/CPSC503\\_Project\\_Report\\_Tianyu.pdf](http://www.cs.ubc.ca/~carenini/TEACHING/CPSC503-14/FINAL-REPORTS-10/CPSC503_Project_Report_Tianyu.pdf) > (Mar. 21, 2015).
- Loch-Dehbi, S., and Plumer, L. (2011). "Automatic reasoning for geometric constraints in 3D city models with uncertain observations." *ISPRS J. Photogrammetry and Remote Sensing*, 66(2011), 177-187.

- Lu, W. (2014). "Semantic parsing with relaxed hybrid trees." *Proc., the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Stroudsburg, PA, 1308-1318.
- Magnini, B., Negri, M., and Tanev, H. (2002). "Is it the right answer? exploiting web redundancy for answer validation." *Proc., 40th Annual Meeting of the Association for Comput. Linguistics*, Association for Computational Linguistics, Stroudsburg, PA, 425-432.
- Mahfouz, T., and Kandil, A. (2012). "Litigation outcome prediction of differing site condition disputes through machine learning models." *J. Comput. Civ. Eng.*, 26(3), 298-308.
- Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (1999) "Performance measures for information extraction." *Proc., DARPA Broadcast News Workshop*, Morgan Kaufmann, San Francisco, California.
- Malik, S.K., Prakash, N., and Rizvi, S. (2010). "Semantic annotation framework for intelligent information retrieval using KIM architecture." *Int. J. Web & Semantic Tech. (IJWest)*, 1(4), 12-26.
- Manicassamy, J., Dhavachelvan, P., and Baskaran, R. (2012). "Metrics based quality assessment for retrieval ability of web-based bioinformatics tools." *Advances in Computer Science, Eng. & Appl., AISC 166*, Springer-Verlag, Berlin Heidelberg, Germany, 947-958.
- Manning, C., Raghavan, P., and Shutze, H. (2009). "An introduction to information retrieval." Cambridge University Press, Cambridge, England.
- Marcinczuk, M., and Ptak, M. (2012). "Preliminary Study on Automatic Induction of Rules for Recognition of Semantic Relations between Proper Names in Polish Texts." *LNCS*, 7499, 264-271.
- Marquez, L. (2000). "Machine learning and natural language processing." *Conf., "Aprendizaje Automatico Aplicado Al Procesamiento Del Lenguaje Natural"*, Soria, Spain.
- Martins, J.P., and Monteiro, A. (2013). "LicA: A BIM based automated code-checking application for water distribution systems." *Autom. Constr.*, 29(2013), 12-23.
- Maynard, D., Bontcheva, K., and Cunningham, H. (2004). "Automatic language-independent induction of gazetteer lists." *Proc., 4th Language Resources and Evaluation Conf. (LREC'04)*, Lisbon, Portugal.

- Maynard, D., Peters, W., and Li, Y. (2006). "Metrics for evaluation of ontology-based information extraction." *WWW 2006 Workshop "Evaluation of Ontologies for the Web"*, Edinburgh, UK.
- McCallum, A., Bellare, K., and Pereira, F. (2005). "A conditional random field for discriminatively-trained finite-state string edit distance." *Proc., the Uncertainty in AI Conf.*, Edinburgh, UK.
- Meystre, S.M., and Haug, P.J. (2005). "Comparing natural language processing tools to extract medical problems from narrative text." *AMIA Annual Symposium Proceedings*, 525-529.
- Mignard, C., and Nicolle, C. (2014). "Merging BIM and GIS using ontologies application to urban facility management in ACTIVE3D." *Computers in Industry*, 65(2014), 1275-1290.
- Moens, M. (2006). "Information extraction: algorithms and prospects in a retrieval context." *Comput.. Linguistics*, 34(2), 315-317.
- Müller H.M., Kenny E.E., and Sternberg P.W. (2004). "Textpresso: an ontology-based information retrieval and extraction system for biological literature." *PLoS Biol.*, 2(11), 1984-1998.
- Nadeau, D., and Sekine, S. (2007). "A survey of named entity recognition and classification." *Linguisticae Investigationes*, 30(1), 3-26.
- Nawari, N.O. (2011). "Automating codes conformance in structural domain." *Proc., Comput. Civ. Eng.*, ASCE, Reston, VA, 569-577.
- National Institute of Building Sciences, (2014). "National BIM standard – United States Version 2." <<http://www.nationalbimstandard.org/faq.php#faq1>> (Jun. 17, 2014).
- Newcombe, R.G. (1998). "Two sided confidence intervals for the single proportion: comparison of seven methods." *Statist. Med.*, 17, 857-872.
- Nguyen,T., and Kim, J. (2011). "Building code compliance checking using BIM technology." *Proc., 2011 Winter Simulation Conference (WSC)*, IEEE, New York, NY, 3395-3400.
- Niemeijer, R.A., Vries, B.D., and Beetz, J. (2009). "Check-mate: automatic constraint checking of IFC models." In A Dikbas, E Ergen & H Giritli (Eds.), *Manag. IT in Constr. Manag. Constr. for Tomorrow*, CRC Press, London, UK, 479-486.
- Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J.R. (2012). "Learning

- multilingual named entity recognition from Wikipedia.” *Artificial Intelligence*, <http://dx.doi.org/10.1016/j.artint.2012.03.006>.
- Noy, N.F., and Hafner, C.D. (1997) “The state of the art in ontology design: a survey and comparative review.” *AI Magazine*, 18(3), 53-74.
- Olson, D.L., and Delen, D. (2008). “Advanced Data Mining techniques (1st ed.)” Springer Publishing Company, Incorporated, New York, NY.
- Open IFC Tools. (2010). <[http://www.openifctools.org/Open\\_IFC\\_Tools/Home.html](http://www.openifctools.org/Open_IFC_Tools/Home.html)> (Jun. 15, 2015).
- OptaSoft, (2014). “International Code Council.” <<http://www.optasoft.com/icc.html>> (Jan. 06, 2014).
- Oracle. (1999). “Essentials of the Java programming language, Part 1.” <<http://www.oracle.com/technetwork/java/index-138747.html>> (Jun. 25, 2015).
- Oracle. (2015). “Java SE downloads.” <<http://www.oracle.com/technetwork/java/javase/downloads/index.html>> (Jun. 17, 2015).
- Orna-Montesinos, C. (2010). “Hyponymy relations in construction textbooks: a corpus-based analysis.” *Linguistic and Translation Studies in Scientific Communication, Linguistic Insights*, 86, 96-114.
- Park, S., and Koo, K. (2011). “CBR-Genetic Algorithm based design team selection model for large-scale design firms.” *KSCE J. Civ. Eng.*, 15(7), 1141-1148.
- Pan, J., Cheng, C.J., Lau, G.T., and Law, K.H. (2008). “Utilizing statistical semantic similarity techniques for ontology mapping - with applications to AEC standard models.” *Tsinghua Sci. and Technol.*, 13(S1), 217-222.
- Pasca, M. (2011). “Attribute extraction from synthetic web search queries.” *Proc., 5th Int. Joint Conf. Natural Language Processing*, Chiang Mai, Thailand, 401-409.
- Patwardhan, S. (2010). “Widening the field of view of information extraction through sentential event recognition.” Ph.D. Thesis, University of Utah, Salt Lake City, UT.
- Pauwels, P., Van Deursen, D., Verstraeten, R., Roo, J.D., Meyer, R.D., Walle, R.V.D., and Campenhout, J.V. (2011). “A semantic rule checking environment for building performance checking.” *Autom. Constr.*, 20(2011), 506-518.

- Pilan, I., Volodina, E., and Johansson, R. (2014). "Rule-based and machine learning approach for second language sentence-level readability." *Proc., the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, Stroudsburg, PA, 174-184.
- Plake, C., Hakenberg, J., and Leser, U. (2005). "Learning patterns for information extraction from free text." *Proc., Workshop des Arbeitskreises Knowledge Discovery.*, Karlsruhe, Germany.
- Poibeau, T., and Messiant, C. (2008). "Do we still need gold standards for evaluation?" *Proc., the Language Resources and Evaluation Conference (LREC)*, ELRA, 547-552.
- Poole, D., and Mackworth, A. (2010). "Artificial intelligence: foundations of computational agents." Cambridge University Press, Cambridge, United Kingdom.
- Popescu, A. (2007). "Information extraction from unstructured web text." Ph.D. thesis, University of Washington, Seattle, WA.
- Porter, M. (1980). "An algorithm for suffix stripping." *Program (Autom. Libr. and Inf. Syst.)*, 14(3), 130-137.
- Portoraro, F. (2011). "Automated reasoning." *The Stanford Encyclopedia of Philosophy (Summer 2011 Edition)*, Edward N. Zalta (ed.) <<http://plato.stanford.edu/archives/sum2011/entries/reasoning-automated/>> (May 20, 2013).
- Pradhan, S., Ward, W., Hacioglu, K., Martin, J.H., and Jurafsky, D. (2004). "Shallow semantic parsing using support vector machines." *Language of Colorado*, Boulder, CO.
- Python v2.7.3 [Computer software]. Beaverton, OR, Python Software Foundation.
- Raikwal, J.S., and Saxena, K. (2012). "Performance evaluation of SVM and k-nearest neighbor algorithm over medical data set." *International Journal of Computer Applications*, 50(14), 35-39.
- Rasdorf, W.J., and Lakmazaheri, S. (1990). "A logic-based approach for processing design standards." *Artif. Intell. Eng. Des. Anal. Manuf.*, 4(3), 179-192.
- Rathod, S.G. (2014). "Machine translation of natural language using different approaches: ETSTS (English to Sanskrit Translator and Synthesizer)." *Int. J. Comput. Appl.*, 102(15), 26-31.
- Resnik, P. (1995). "Using information content to evaluate semantic similarity in a taxonomy." *Proc., IJCAI'95*, IJCAI, Inc., Somerset, NJ, 448-453.

- Riloff E. and Lorenzen J. (1999). "Extraction-based text categorization: generating domain-specific role relationships automatically." In *"Natural Language Information Retrieval"*, T. Strzalkowski, ed., Kluwer Academic Publishers, Dordrecht, 167-195.
- Robinson, A., and Voronkov, A. (2001). "Handbook of automated reasoning - Volume 1." The MIT Press, Cambridge, MA.
- Rodríguez, M.A., and Egenhofer, M.J. (2003). "Determining semantic similarity among entity classes from different ontologies." *IEEE T. Knowl. Data En.*, 15(2), 442-456.
- Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychol. Rev.*, 65(6).
- Ross, T.J., Savage, S.J., and Carson, J.M. (1990). "DAPS: expert system for structural damage assessment." *J. Comput. Civ. Eng.*, 4(4), 327-348.
- Roth-Berghofer, T., Adrian, B., and Dengel, A. (2010). "Case acquisition from text: ontology-based information extraction with SCOOBIE for myCBR." *Proc., ICCBR'2010*, 451-464.
- Rubino, R., Rotolo, A., and Sartor, G. (2004). "An OWL ontology of norms and normative judgements." The European project for Standardized Transparent Representations in order to Extend Legal Accessibility.
- Ryu, Y.U., and Lee, R.M. (1995). "Defeasible deontic reasoning and its applications to normative systems." *Decision Support Systems*, 14(1), 59-73.
- Saggion, H., Funk, A., Maynard, D., and Bontcheva, K. (2007). "Ontology-based information extraction for business intelligence." *Proc., 6th Int. The semantic Web and 2nd Asian Conf., Asian Semantic Web (ISWC'07/ASWC'07)*, Springer-Verlag, Berlin, Heidelberg, Germany, 843-856.
- Saint-Dizier, P. (1994). "Advanced logic programming for language processing." Academic Press, San Diego, CA.
- Salama, D.M., and El-Gohary, N.M. (2013a). "Semantic text classification for supporting automated compliance checking in construction." *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000301.
- Salama, D.M., and El-Gohary, N.M. (2013b). "Automated compliance checking of construction operation plans using a deontology for the construction domain." *J. Comput. Civ. Eng.*, 27(6), 681-698.



- Salama, D.M., and El-Gohary, N.M. (2011). "Semantic modeling for automated compliance checking." *Proc., Congress on Comput. Civ. Eng.*, ASCE, Reston, VA, 641-648.
- Santorini, B. (1990). "Part-of-speech tagging guidelines for the Penn Treebank project (3rd Revision, 2nd printing). <<https://catalog.ldc.upenn.edu/docs/LDC99T42/tagguid1.pdf>> (Apr. 30, 2015).
- Santos, I.A., and Farinha, F. (2005). "Code checking automation in building design: new trends for cognition." *Proc., 2005 ASCE Int. Conf. Comput. Civ. Eng.*, 179(14).
- Sapkota, K., Aldea, A., younas, M., Duce, D.A., and Banares-Alcantara, R. (2012). "Extracting meaningful entities from regulatory text." *2012 Fifth IEEE Int. Workshop Requirements Eng. and Law (RELAW)*, IEEE, Piscataway, NJ, 29-32.
- Saric, J., Jensen, L.J., and Rojas, I. (2005). "Large-scale extraction of gene regulation for model organisms in an ontological context." *Silico Bio.*, 5(2005), 21-32.
- Selvi, R., Saravan Kumar, S., and Suresh, A. (2015). "An intelligent intrusion detection system using average manhattan distance-based decision tree." L.P. Suresh et al. (eds.), *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, Advances in Intelligent Systems and Computing 324*.
- Shehata, S. (2009). "A WordNet-based semantic model for enhancing text clustering." *2009 IEEE Int. Conf. Data Mining. Workshops*, IEEE, Piscataway, NJ, 477-482.
- Simpson, T., and Dao, T. (2010). "WordNet-based semantic similarity measurement." <<http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement>> (Aug. 14, 2014).
- Singapore Building and Construction Authority. (2006). "Construction and real estate network: Corenet Systems." <<http://www.corenet.gov.sg/>> (Jul. 15, 2011).
- Sinha, S., Sawhney, A., Borrmann, A., and Ritter, F. (2013). "Extracting information from building information models for energy code compliance of building envelope." *COBRA 2013 Conf.*, International Council for Research and Innovation in Building and Construction (CIB), Rotterdam, Netherlands.
- Slimani, T. (2013). "Description and evaluation of semantic similarity measures approaches." *Int. J. Comput. Appl.*, 80(10), 25-33.
- Song, W., Liang, J.Z., and Park, S.C. (2014). "Fuzzy control GA with a novel hybrid semantic similarity strategy for text clustering." *Inform. Sciences*, 273(2014), 156-170.

- Soysal, E., Cicekli, I., and Baykal, N. (2010). "Design and evaluation of an ontology based information extraction system for radiological reports." *Comput. Bio. Medicine*, 40(11-12), 900-911.
- Spiliopoulos, V., Vouros, G.A., and Karkaletsis, V. (2010). "On the discovery of subsumption relations for the alignment of ontologies." *Web Semant.*, 182, 1-20.
- Slimani, T. (2013). "Description and evaluation of semantic similarity measures approaches." *Int. J. Comput. Appl.*, 80(10), 25-33.
- Steel, J., Drogemuller, R., and Toth, B. (2010). "Model interoperability in building information modeling." *Software and Systems Modeling*, 2010, 1-12.
- Stenmark, M., and Malec, J. (2014). "Knowledge-based instruction of manipulation tasks for industrial robotics." *Robotics and Computer-Integrated Manufacturing*, <http://dx.doi.org/10.1016/j.rcim.2014.07.004i>.
- Sterling, L., and Shapiro, E. (1986). "The art of Prolog: advanced programming techniques." Massachusetts Institute of Technology, Cambridge, MA.
- Suchanek, M., Kasneci, G., and Weikum, G. (2007). "YAGO: A core of semantic knowledge unifying WordNet and Wikipedia." *Proc., WWW 2007*, Association for Computing Machinery, New York, NY, 697-706.
- Sudha, L.R., and Bhavani, R. (2012). "Performance comparison of SVM and kNN in automatic classification of human gait patterns." *International Journal of Computers*, 1(6), 2012.
- Sun, A., Grishman, R., and Sekine, S. (2011). "Semi-supervised relation extraction with large-scale word clustering." *Proc., 49th Annual Meeting of the Assoc. for Comput. Linguistics: Human Language Technologies - Volume 1 (HLT '11)*, Assoc. for Comput. Linguistics, Stroudsburg, PA, 521-529.
- Tang, K., Li, F., and Daphne, K. (2012). "Learning latent temporal structure for complex event detection." *Proc., CVPR. 2012*, IEEE, New York, NY, 1250-1257.
- Tan, X., Hammad, A., and Fazio, P. (2010). "Automated code compliance checking for building envelope design." *J. Comput. Civ. Eng.*, 24(2), 203-211.
- Tan, X., Hammad, A., and Fazio, P. (2007). "Automated code compliance checking of building envelope performance." *Proc., 2007 ASCE Int. Workshop on Comput. Civ. Eng.*, ASCE, Reston, VA, 256-263.
- Teicholz, P. (2013). "BIM for facility managers." John Wiley & Sons, Inc., Hoboken, New Jersey.

- Tetreault, J.R., and Chodorow, M. (2008). "Native judgments of non-native usage: experiments in preposition error detection." *COLING Workshop on Human Judgments in Computational Linguistics*, Association for Computational Linguistics Stroudsburg, PA.
- Tierney, P.J. (2012). "A qualitative analysis framework using natural language processing and graph theory." *The Intl. Review of Research in Open and Distance Learning*, 13(5).
- Torma, S. (2015). "Web of building data – integrating IFC with the Web of Data." *eWork and eBusiness in Architecture, Engineering and Construction – Martens, Mahdavi & Scherer (Eds)*, Taylor & Francis Group, London, 141-147.
- Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). "Feature-rich part-of-speech tagging with a cyclic dependency network." *Proc., HLT-NAACL 2003*, 252-259.
- Turney, P.D., and Littman, M. (2003). "Measuring praise and criticism: inference of semantic orientation from association." *ACM Transactions on Information Systems*, 21(4), 315-346.
- Turney, P.D. (2002). "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews." *Proc., 40th Annual Meeting of the Association for Comput. Linguistics*, Association for Computational Linguistics, Stroudsburg, PA, 129-159.
- Udaipurwala, A. and Russell, A. D. (2000). "Reasoning about construction methods." *Proc., Construction Congress VI*, Orlando, Florida, 386-395.
- Ugwu, O.O., Arcisweski, T., and Anumba, C.J. (2002). "Teaching agents how to solve design problems: a mixed initiative learning strategy." *Proc., Int. Workshop Information Tech. Civ. Eng.*, Washington, D.C., 11-24.
- University of Sheffield. (2013). "General architecture for text engineering." <<http://gate.ac.uk/>> (Oct. 13, 2013).
- US Department of Energy (US DOE). (2013). *REScheck: Residential Compliance Using REScheck*. <<http://www.energycodes.gov/rescheck>> (Jul. 11, 2013).
- US Department of Energy (US DOE). (2011). *Building energy codes program: Software and tools*. <<http://www.energycodes.gov/software.stm>> (Jul. 15, 2011).
- US Environmental Protection Agency (US EPA). (2004). "Wal-Mart II storm water settlement." *Civil Enforcement Cases and Settlements*, <<http://www.epa.gov/compliance/resources/cases/civil/cwa/walmart2.html>> (Nov. 25, 2012).

- Uschold, M. and King, M. (1995) "Towards a methodology for building ontologies." *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada.
- Van Rijsbergen, C. J. (1979). "Information retrieval." Information Retrieval Group, University of Glasgow, Glasgow, Scotland.
- Vanlande, R., Nicolle, C., and Cruz, C. (2008). "IFC and building lifecycle management." *Autom. Constr.*, 18(1), 70-78.
- Varelas, G., Voutsakis, E., and Raftopoulou, P. (2005). "Semantic similarity methods in WordNet and their application to information retrieval on the web." *Proc., 7th annual ACM intl. workshop on Web inform. and data manage. (WIDM '05)*, Association for Computing Machinery, New York, NY, 10-16.
- Vo, D., Hai, V.T., and Ock, C. (2015). "Exploiting language models to classify events from twitter." *Computational Intelligence and Neuroscience*, 401024.
- Vrandećić, D. (2010). "Ontology evaluation." Ph.D. Thesis, the Karlsruhe Institute for Technology (KIT), Karlsruhe, Germany.
- Vries, B.D., and Steins, R.J. (2008). "Assessing working conditions using fuzzy logic." *Autom. Constr.*, 17(2008), 584-591.
- Wagner, E., Liu, J., Birnbaum, L., Forbus, K.D., and Baker, James. (2006). "Using explicit semantic models to track situations across news articles." *Proc., 2006 AAAI Workshop Event Extraction and Synthesis*, AAAI, Palo Alto, California, 42-47.
- Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y., and Guo, J. (2010). "Mining advisor-advisee relationships from research publication networks." *Proc., 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '10)*, ACM, New York, NY, 203-212.
- Wang, X., Guo, L., and Fang, J. (2008). "Automated ontology selection based on description logic." *12th Int. Conf. Comput. Supported Cooperative Work in Design*, Xi'an, China, 482-487.
- Weldu, Y.W., and Knapp, G.M. (2012). "Automated generation of 4D building information models through spatial reasoning." *Proc., 2012 ASCE Constr. Research Congress (CRC)*, West Lafayette, IN.
- Wen, D., Cuzzola, J., Brown, L., and Kinshuk. (2012). "Exploiting semantic roles for asynchronous question answering in an educational setting." *Canadian Artif. Intel. 2012, LNAI 7310*, Springer-Verlag, Berlin Heidelberg, Germany, 374-379.

- Wiebe, J., Wilson, T., and Cardie, C. (2005). "Annotating expressions of opinions and emotions in language." *Language Resources and Evaluation*, 39(203), 165-210.
- Wilson, E.B. (1927). "Probable inference, the law of succession, and statistical inference." *Journal of the American Statistical Association*, 22(158), 209-212.
- Wimalasuriya, D.C., and Dou, D. (2010). "Components for information extraction: ontology-based information extractors and generic platforms." *J. Information Sci.*, 36(3), 306-323.
- Windisch, R., Wulfing, A., and Scherer, R.J. (2012). "A generic filter concept for the generation of BIM-based domain- and system-oriented model views." *eWork and eBusiness in Architecture, Engineering and Construction*, Taylor & Francis Group, London, UK.
- Wissler, L, Almashraee, M., Monett, D., and Paschke, A. (2014). "The gold standard in corpus annotation." <  
[http://www.ieee-student-conference.de/fileadmin/papers/2014/ieeegsc2014\\_submission\\_3.pdf](http://www.ieee-student-conference.de/fileadmin/papers/2014/ieeegsc2014_submission_3.pdf)> (May 28, 2015).
- Wyner, A., and Governatori, G. (2013). "A study on translating regulatory rules from natural language to defeasible logic." *Proc., RuleML 2013:7th Int. Web Rule Symp.*, Springer, Berlin.
- Wyner, A., and Peters, W. (2011). "On rule extraction from regulations." *Proc., JURIX 2011: 24th Int. Conf. Legal Knowledge and Information-Systems*, IOS Press, Amsterdam, Netherlands, 113-122.
- Yang, D., and Eastman, C. (2007). "A rule-based subset generation method for product data models." *Comput.-Aided Civ. Infra. Eng.*, 22(2007), 133-148.
- Yin, S., and Fan, G. (2013). "Research of POS tagging rules mining algorithm." *Appl. Mech. Mater.*, 347-350, 2836-2840.
- Young, N.W., Jr., Jones, S.A., Bernstein, H.M., and Gudgel, J.E. (2009). "The business value of BIM: getting building information modeling to the bottom line." The McGraw-Hill Companies, New York, NY.
- Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., and Zarli, A. (2010). "Adaptation of the domain ontology for different user profiles: application to conformity checking in construction." *Web Information Systems and Technologies, Lecture Notes in Business Information Processing*, 45, 128-141.
- Yurchyshyna, A., and Zarli, A. (2009). "An ontology-based approach for formalisation and

- semantic organisation of conformance requirements in construction.” *Autom. Constr.* 18 (8): 1084-1098.
- Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., and Zarli, A. (2008). “Towards an ontology-enabled approach for modeling the process of conformity checking in construction.” *Proc., CAiSE’08 Forum 20th Intl. Conf. Adv. Info. Sys. Eng.*, dblp team, Germany, 21-24.
- Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., Lima, C, and Zarli, A. (2007). “Towards an ontology-based approach for conformance checking modeling in construction.” *Proc., CIB W78 2007*, Conseil International du Bâtiment (CIB), Rotterdam, Netherlands.
- Zhai, H., Lingren, T., Deleger, L., Li, Q., Kaiser, M., Stoutenborough, L., and Solti, I. (2013). “Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing.” *J. Med. Internet. Res.*, 15(4), e73.
- Zhang, J., and El-Gohary, N.M. (2013). “Handling sentence complexity in information extraction for automated compliance checking in construction.” *Proc., CIB W78 2013*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.
- Zhang, J., and El-Gohary, N.M. (2011). “Automated information extraction from construction-related regulatory documents for automated compliance checking.” *Proc., CIB W78 2011*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.
- Zhang, L., and Issa, R.R.A., (2013). “Ontology-based partial building information model extraction.” *J. Comput. Civ. Eng.*, 27(6), 576-584.
- Zhang, S., Teizer, J., Lee, J., Eastman, C.M., and Venugopal, M. (2013). “Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules.” *Autom. Constr.*, 29(2013), 183-195.
- Zhong, B.T., Ding, L.Y., Luo, H.B., Zhou, Y., Hu, Y.Z., and Hu, H.M. (2012). “Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking.” *Autom. Constr.*, 28(2012), 58-70.
- Zhou, N. (2012). “B-Prolog user’s manual (version 7.7), Prolog, agent, and constraint programming.” *Afany Software*. <<http://www.probp.com/manual/manual.html>> (Nov. 19, 2012).
- Zhou, P., and El-Gohary, N. (2014). “Ontology-based multi-label text classification for enhanced information retrieval for supporting automated environmental compliance checking.” *Proc., 2014 ASCE Constr. Res. Congress (CRC)*, ASCE, Reston, VA, 2238-2245.

- Zhu, X., Wang, H., Gan, H., and Gao, C. (2011). "Construction and management of automatical reasoning supported data mining metadata." *2011 Int. Conf. Business Manage. Electronic Information (BMEI)*, Guangzhou, China, 205-210.
- Zouaq, A. (2011). "Ontology learning and knowledge discovery using the web." *An Overview of Shallow and Deep Natural Language Processing for Ontology Learning*, IGI Global., Hershey, PA, 16-38.

## 11 APPENDIX A: ANNOTATION GUIDELINES FOR REGULATORY INFORMATION EXTRACTION

The scheme focuses on the semantic information elements of quantitative requirements. In each requirement sentence: (1) there should be one and only one instance of each of subject, comparative relation, quantity value, quantity unit, and quantity reference; (2) there should be at most one instance of each of compliance checking attribute, deontic operator indicator, and quantitative relation; and (3) there could be zero, one, or more instances of each of subject restriction and quantity restriction.

### 1. Subject

This semantic information element represents a “thing” (e.g., building object, space) that is subject to a particular regulation or norm.

*(Subject: Courts) shall not be less than 3 feet in width.*

### 2. Compliance checking attribute

This semantic information element represents a specific characteristic of a “subject” by which its compliance is assessed.

*Yards shall not be less than 3 feet in (Compliance checking attribute: width) for one and two story buildings.*

### 3. Deontic operator indicator

This semantic information element represents a deontic modal operator applicable to the current requirement. There are three types: obligation, permission, and prohibition.

*Habitable spaces, other than a kitchen, (Deontic operator indicator: shall) not be less*



*than 7 feet in any plan dimension.*

#### 4. Quantitative relation

This semantic information element represents the type of relation for the quantity.

*The court shall be (Quantitative relation: increased) 1 foot in width and 2 feet in length for each additional story.*

#### 5. Comparative relation

This semantic information element represents a relation that is commonly used to compare quantitative values. There are five types: greater than or equal, less than or equal, greater than, less than, and equal.

*Occupiable spaces, habitable spaces and corridors shall have a ceiling height of (Comparative relation: not less than) 7 feet 6 inches.*

#### 6. Quantity value

This semantic information element represents a value or a range of values that defines the quantified requirement.

*Every dwelling unit shall have at least one room that shall have not less than (Quantity value: 120) square feet of net floor area.*

#### 7. Quantity unit

This semantic information element represents the unit of measure for the quantity value.

*The unit shall have a living room of not less than 220 (Quantity unit: square feet) of floor area.*

8. Quantity reference

This semantic information element represents a reference to another quantity (which presumably includes a value and a unit).

*The minimum openable area to the outdoors shall be 4 percent of the (Quantity reference: floor area being ventilated).*

9. Subject restriction

This semantic information element places a constraint on the definition of a subject.

*Courts (Subject restriction: having windows opening on opposite sides) shall not be less than 6 feet in width.*

10. Quantity restriction

This semantic information element places a constraint on the definition of a quantity.

*The minimum net area of ventilation openings shall not be less than 1 square foot (Quantity restriction: for each 150 square feet of crawl space area).*

## 12 APPENDIX B: ANNOTATION GUIDELINES FOR REGULATORY INFORMATION TRANSFORMATION

The scheme focuses on the concept and relation logic clause elements of quantitative requirements. The goal of the annotation is to identify the names and arguments of all concept logic clause elements and relation logic clause elements in a quantitative requirement. Each instance of a concept logic clause is represented by a predicate with one argument. Each instance of a relation logic clause is represented by a predicate with one or more arguments.

### 1. Concept logic clause

A concept logic clause represents a concept. A concept is expressed using a noun or noun phrase. The determiners are not included in a concept logic clause. Quantity values and quantity units are not included in a concept logic clause either. The identified noun or noun phrase is used both as the name and the argument of a concept predicate.

*(Concept: Courts) shall not be less than 3 feet in (Concept: width).*

*(Concept: Occupiable spaces), (Concept: habitable spaces) and (Concept: corridors) shall have a (Concept: ceiling height) of not less than 7 feet 6 inches.*

### 2. Relation logic clause

A relation logic clause represents a relation. There are four types of relation logic clauses: relation logic clause with one argument, relation logic clause with two arguments, relation logic clause with multiple arguments, and compound relation logic clause.

2.1 A relation logic clause with one argument represents a description of a concept.

*The minimum openable area to the outdoors shall be 4 percent of the (Relation: floor area being ventilated).*

*Predicate name: ventilated*

*Predicate argument: floor area*

2.2 A relation logic clause with two arguments represents a relation between two concepts.

*(Relation: Kitchens shall have a clear passageway) of not less than 3 feet between counter fronts and appliances or counter fronts and walls.*

*Predicate name: have*

*Predicate arguments: kitchens, clear passageway*

2.3 A relation logic clause with multiple arguments represents a relation between three or more concepts.

*A minimum of 1 inch of (Relation: airspace shall be provided between the insulation and the roof sheathing).*

*Predicate name: between*

*Predicate arguments: airspace, insulation, roof sheathing*

2.4 A compound relation logic clause represents a relation that uses embedded predicates.

*Courts shall (Relation: not be less than 3 feet in width).*

*Predicate name: not less than*

*Predicate arguments: width, quantity(3, feet)*