ITcon

www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

A KNOWLEDGE REPRESENTATION APPROACH IN BIM RULE REQUIREMENT ANALYSIS USING THE CONCEPTUAL GRAPH

SUBMITTED: March 2016
REVISED: June 2016
PUBLISHED: November 2016 at http://www.itcon.org/2016/24
GUEST EDITORS: Dimyadi J. & Solihin W.

Wawan Solihin
School of Architecture, Georgia Institute of Technology, USA;
wawan.solihin@gatech.edu

Charles Eastman, Professor
School of Architecture, Georgia Institute of Technology, USA;
charles.eastman@coa.gatech.edu

SUMMARY: *Specifications of the exact requirements for BIM-based rules are often much more complex than it appears from their direct language interpretation. It is because the complete rule interpretation involves knowledge from human experts. Therefore, detailed interpretation process of rules for the purpose of automated rule checking implementation must be able to capture such knowledge and it must be retained throughout the implementation process. In order to ensure both completeness and precision, the knowledge capture needs to be formalized. We propose a formalized way to capture these requirements using a knowledge representation approach. Detailed analysis of the structure of the rules points to the conceptual graph (CG) as a suitable method for this purpose due to its expressiveness, which allows unambiguous description of the requirements. Such unambiguous description can be understood by all the participants in the implementation efforts. The conceptual graph is a representation that conforms to the first order logic that makes it suitable for the job. Using the conceptual graph, rules can be broken into their atomic rules and incorporates the unwritten domain checking knowledge that if done right will remove ambiguities that often plague building codes. It also provides a standardized way to capture and document the model data requirements and the high level checking logic as their functional requirements. With this, a layer that often separates the rule experts and the implementers can be eliminated, resulting in clarity and immediate usefulness for the implementation. The validity of this assertion is demonstrated using the comparison to the actual corresponding implementation of the rules.*

KEYWORDS: *BIM, Knowledge Representation, Conceptual Graph, Rule Checking*

REFERENCE: *Wawan Solihin, Charles Eastman (2016). A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. Journal of Information Technology in Construction (ITcon), Special issue: CIB W78 2015 Special track on Compliance Checking, Vol. 21, pg. 370-401, http://www.itcon.org/2016/24*

1 INTRODUCTION

One of the important steps in a rule checking implementation process is the rule requirement analysis. This step is unique for building rules, especially building codes, because it involves a step known as a rule interpretation. The aim of the interpretation step is not only to understand the intent of the rule but also to gather knowledge from the rule experts (C Eastman, Lee, Jeong, & Lee, 2009). In many of research papers on this subject, the importance of the interpretation process has not received its due attention. Attention appears to be given only when the effort relates to the real software implementation in which the actual users or experts are involved because the accuracy of checking becomes critical. One example of such cases is the implementation of the automated rule checking system at the national level in Singapore - CORENET ePlanCheck (Khemlani, 2005; novaCITYNETS, 2000; Swaddiwudhipong & Kog, 2000). A Recent project in South Korea may cover a similar scope, but it is still in the early stage of development (Kim, 2015). Based on the experience in implementing CORENET ePlanCheck, the interpretation step can take as much as 30% of the total time to implement a rule. Complex rules typically found in building codes are a combination of several aspects that contribute to their complexity. They involve the language structure, the domain knowledge embedded in the rules, and their logic structure. Added to those technical aspects of the rules is the knowledge of the human experts. The knowledge covers many implicit assumptions within the relevant domain and, more importantly, the unwritten knowledge on how the experts interpret the rules. This is usually an accumulated knowledge over many years of experience dealing with the real world cases. A study by Fiatech confirmed that when human interpretation is involved, inconsistencies are expected. Different officers tend to interpret the rules differently, often colored by their experience and locality (Fiatech, 2012). Some rules in the CORENET ePlanCheck implementation went through multiple iterations and revisions because of the same reason when multiple reviewers are involved. To make the matter worse, software developers who are in charge of translating the semantic knowledge of the rule requirements into computer codes are not the ones directly involved in the rule interpretation.

The role of knowledge capture extends beyond just the interpretation step to the entire rule checking system development. The typical rule checking development process can be described in the diagram shown in Figure 1. It involves multiple actors in different stages of the workflow. Apart of the rule analysts who capture the rule requirements from the rule texts and the human experts for the benefit of the software developers, part of the knowledge should also be accessible to the BIM authoring tool providers for their support of the relevant data exchange requirements. The knowledge also is required by the modeler who will design buildings and submit the design for compliance checks. The requirement for data exchange is an important step that has been recognized. BuildingSMART has defined standard processes to capture the data exchange requirements using IDM (Information Delivery Manual) and MVD (Model View Definition) (C. Eastman, Jeong, Sacks, & Kaner, 2010; Charles Eastman & Sacks, 2010). Whereas the modeler is typically given the modeling guidelines or standards to follow. However, the guidelines are only useful when there are clear specifications on what information must be explicitly in the model for specific purposes or rules.

This paper addresses the issue of formalizing the knowledge capture during the interpretation process for the purpose of documenting the rule requirements using a knowledge-based approach. The aim is to create a consistent representation of the rule requirements and the expert knowledge to minimize knowledge loss when such knowledge flows among the actors involved in the entire process of rule checking system development. This paper is organized in the following manner. First, we will discuss current approaches to capturing the rule requirements. It is followed by a detailed description how this was done in CORENET ePlanCheck project. From here we will begin to look into the semantic representation of the rule requirements as knowledge. In the next section a description how the Conceptual Graph fits the purpose set forth in this paper will be described, followed by discussions on the issue of granularity of the conceptual graph and the potential mapping from the CG to MVD and UML. To show the usefulness of the approach, we validated a few CG with the actual implementation of the rules using the original CORENET ePlanCheck implementation and using our research work on BIM Rule Language (BIMRL) (Wawan, 2016). In this paper, we will focus on IFC data format as the source of BIM data. IFC is a widely used standard in the AECO industry and it has been published as an ISO standard (ISO, 2013).

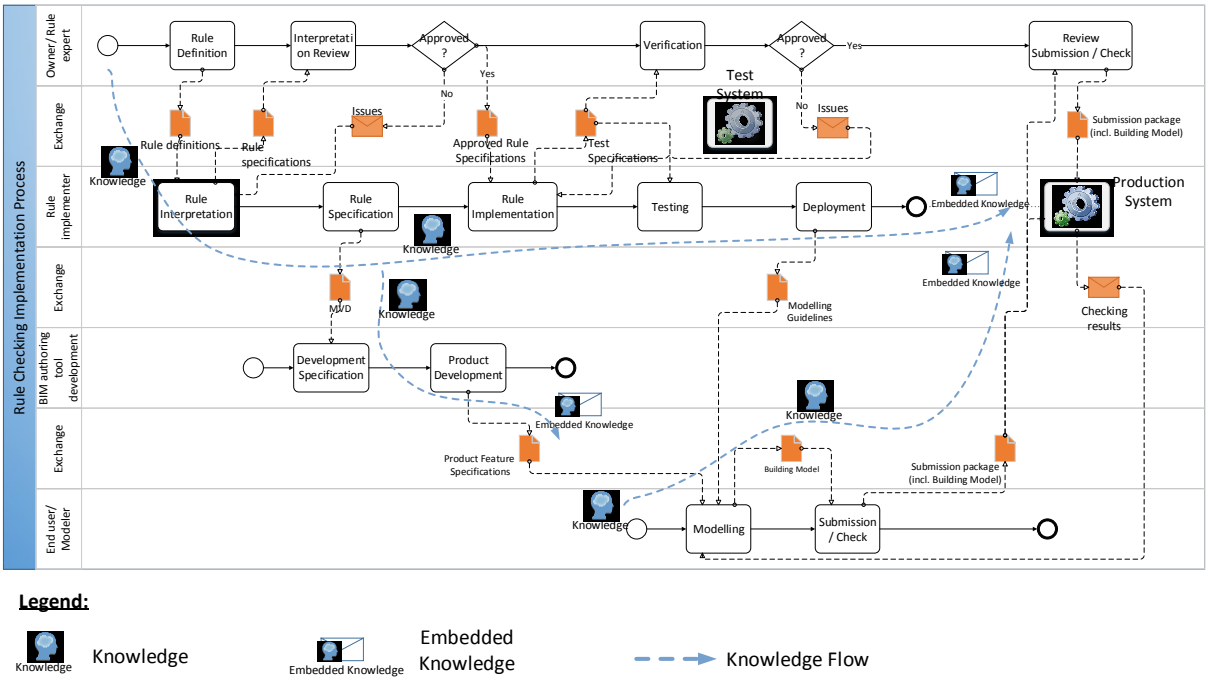


Figure 1 - A Typical Rule Implementation Process and the Knowledge Flow

2 CURRENT APPROACHES TO CAPTURE THE RULE REQUIREMENTS

For many years since rule checking research started in the 1960's with Fenves's introduction of the decision table to define the computable form of building structural codes (Fenves, 1966), many different approaches have been proposed. In general, there are several broad categories with respect to the focus of the approaches. They are semantic rule structure, production rule, logic based implementation, and language driven.

Among efforts with a focus on the semantic rule structure is the RASE methodology that tags the rules based on four classifications - Requirement, Applicability, Selection, Exception (Beach, Kasim, Li, Nisbet, & Rezgui, 2013; Hjelseth & Nisbet, 2011). Since RASE methodology only deals with the semantic rule structure, it requires another method for the rule definition and execution. Using RASE approach, several implementation methods have been proposed including using the IFC constraint schema (Hjelseth & Nisbet, 2010) and combining it with rule engine (Beach et al., 2013). Another more recent trend that is gaining popularity is the semantic web approach using OWL and RDF graph (Bouzidi, Fies, Faron-Zucker, Zarli, & Thanh, 2012; Pauwels et al., 2011; Yurchyshyna & Zarli, 2009). In this approach, model data is translated into RDF triplets and stored into the RDF graph. Using SPARQL as the query language, rules can be defined by interrogating the graph. Rules potentially can be coded into RDF too. This theoretically will make the rule execution to be straightforward because both data and rules share the same graph and thus language (C. Zhang & Beetz, 2015). There is also a Natural Language Processing approach that has been proposed (Salama & El-Gohary, 2013; J. Zhang & El-Gohary, 2012). These methods suffer from constraint to only explicit data inside the IFC model, and to the explicit terms that can be identified in the language of the rules. This restricts the usefulness of the approach to only very few straightforward rules dealing with the individual objects and their properties and explicit relationships. According to (Solihin & Eastman, 2015), this approach mainly is able to handle class-1 of building rules. This may change in the future if the 3D spatial query is supported in SPARQL in future. OGC currently has published a geographic query language for RDF data called GeoSPARQL (OGC, 2012). It supports only 2D data for GIS related queries.

The second approach using production rule seems intuitively fits well with the rule checking since it deals with rules after all. Each rule in the most basic form should follow the following pattern: **if *<condition>* then *<action>***, which is the essence of the production rule. In this approach, rules will need to be broken into their atomic rules. A rule engine will be used to perform inference from the database of rules based on the facts being fed into the engine from the building model. In general, it is expected that the rule engine provides a true or false answer with respect to whether the fact complies or not compliant with the pre-defined rules. An example of

research papers reporting the use of this approach is found in (Tan, Hammad, & Fazio, 2010). Since the production rules are static and explicit, this approach also suffers from the limitation that it works only with the explicit data. This limitation is similar to the semantic language approach. The production rule approach, however, may allow support for an extended data, but it must be hard-coded to be executable by the rule engine. This usually is restricted only to the well-defined requirements and the numbers are usually limited. It is also not scalable easily to support a wide range of rule requirements from different sub-domains and different countries. The implementation of the built-in extensions usually is a black-box to the user.

The third approach using logic based implementation is used in the commercial software developments, i.e. Solibri and FORNAX (Solibri; Solihin, Shaikh, Rong, & Poh, 2004), and some research works (J. M. Lee, 2010; S. Malsane, J. Matthews, S. Lockley, P. E. Love, & D. Greenwood, 2015). They both employ parametrized but hard-coded rules into the system. This approach is categorized as a black-box approach (Preidel & Borrmann, 2015) since users do not have any idea how the actual checking is done. The advantage of this approach lies in its ability to perform very complex rule without being constrained by another form of expression. It is only limited by its own internal data model, the built-in capability in the system, and as far as a programming language could do. The disadvantage is that it is hard to adapt to changing requirements of the rules.

The last category of approaches is the language-driven approach. BERA language is one of the approaches (J. K. Lee, 2011). Using the domain-specific language approach allows queries to the model to be done and a programming construct to be run on the data in a more transparent way since it is driven by the rule experts themselves. BERA, however, is still lack of generality and it is unable to support complex rule structure. The language itself is also not extensible without modification to the syntax. The main weakness, however, is that BERA develops its own executing environment, which limits its capability to support more complex logic in the complex rules. Recently, there is a preliminary work on using visual language (Preidel & Borrmann, 2015). This approach strikes a balance between a need not to write a program and to support more complex rule logic. It may be a promising approach in future provided that the toolset is rich enough to support the rule requirements.

In all the above discussions, the approaches focus on translating the rules to the computable software modules. Although some papers recognized the need to translate knowledge into computable forms including the expert knowledge, they rarely deal with the need to formalize the rules and their logic requirements for the purpose of documentation and communication. Only papers or projects that deal with the actual implementation beyond the proof-of-concept realizes the importance of capturing the knowledge into a form that is effective for knowledge retention, flow, and communication. This concern is, in fact, a typical software development activities. The unique issue one has to deal with when developing a rule checking system is the unique challenge managing a large number of rules, capturing knowledge from years of experience from the rule experts, and translating those into computable forms. In the next section, we present a brief description of the process done in developing CORENET ePlanCheck with the focus on the knowledge flow.

3 CORENET EPLANCHECK DEVELOPMENT PROCESS

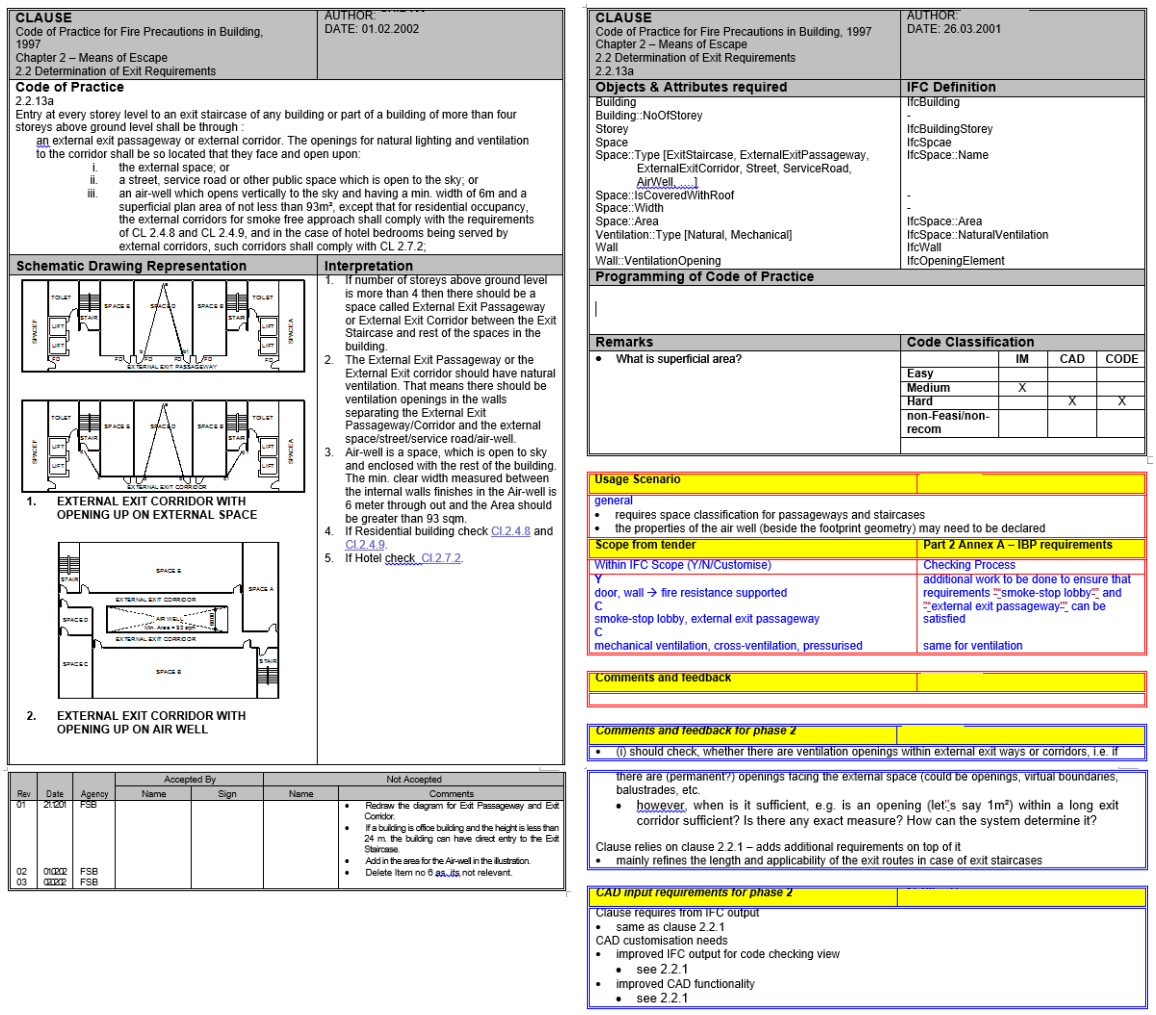
CORENET ePlanCheck project followed a similar process shown in Figure 1. Relevant to this paper, we will focus on the knowledge flow. The knowledge mainly originated from the codes and the owners, through the interpretation process. The process involved interviews and documentation of each code in scope. It resulted in detailed analysis of the rules that are captured in an extensive documentation. Much of the documentation captures more detailed descriptions or explanation of the rules, the checking requirements, the objects in focus for checking, explanations of implicit assumptions and dependencies, checking logic, and the rule exceptions. Figure 2 shows a sample of such documentation. The documentation contains several sections:

a. Rule header section, which described the rule number from the original source and the author of the document plus the date when the document was created.
b. Interpretation section. In the interpretation section, diagrams or floor plans are often used in the left column to aid understanding of the clause and to give the context to which the rule applies. On the right-hand side, the text describing the interpretation is usually listed in a numbered list. The description may contain a clarification of terms, checking conditions, checking logic, constraints and any other knowledge obtained from interviews with the owner of the rule.
c. Interpretation tracking section and approval or acceptance of the interpretation document.
d. An MVD-like section. It is called an MVD-like section since it was done prior to the definition of MVD. It captures what required to be in the IFC model, this is it in essence what MVD is. It also contains an early assessment of the level of difficulty.

The sections below (highlighted in yellow in the document) are internally used within the implementation team:

e. Usage scenario section. It is being filled by the domain expert to assist the development team in understanding the requirement, scope, and scenarios for testing.
f. Comments and feedback section.
g. CAD input requirements section, which is intended to be used for the implementer's agreement when this MVD is implemented by the BIM authoring tool vendor.

The sample is arbitrarily chosen because of its completeness to represent the documents and its compactness in size. Many other documents are much longer than this. As these are description-based documentation that has been done by various people, there was difficulty in maintaining a uniform quality of the documents and there were plenty of "lost in translation" cases. These require a certain degree of redundancy to "interpret" the documents at every major step of development whenever handover from one team to the other occurred, for example from the rule analyst to the developer and from the developer to the tester.



CLAUSE
Code of Practice for Fire Precautions in Building, 1997
Chapter 2 – Means of Escape
2.2 Determination of Exit Requirements

AUTHOR:
DATE: 01.02.2002

Code of Practice
2.2.13a
Entry at every storey level to an exit staircase of any building or part of a building of more than four storeys above ground level shall be through :
an external exit passageway or external corridor. The openings for natural lighting and ventilation to the corridor shall be so located that they face and open upon:
i. the external space; or
ii. a street, service road or other public space which is open to the sky; or
iii. an air-well which opens vertically to the sky and having a min. width of 6m and a superficial plan area of not less than 93m², except that for residential occupancy, the external corridors for smoke free approach shall comply with the requirements of CL 2.4.8 and CL 2.4.9, and in the case of hotel bedrooms being served by external corridors, such corridors shall comply with CL 2.7.2;

Schematic Drawing Representation

1. EXTERNAL EXIT CORRIDOR WITH OPENING UP ON EXTERNAL SPACE

2. EXTERNAL EXIT CORRIDOR WITH OPENING UP ON AIR WELL

Interpretation

1. If number of storeys above ground level is more than 4 then there should be a space called External Exit Passageway or External Exit Corridor between the Exit Staircase and rest of the spaces in the building.
2. The External Exit Passageway or the External Exit corridor should have natural ventilation. That means there should be ventilation openings in the walls separating the External Exit Passageway/Corridor and the external space/street/service road/air-well.
3. Air-well is a space, which is open to sky and enclosed with the rest of the building. The min. clear width measured between the internal walls finishes in the Air-well is 6 meter through out and the Area should be greater than 93 sqm.
4. If Residential building check Cl.2.4.8 and Cl.2.4.9.
5. If Hotel check, Cl.2.7.2.

Rev	Date	Agency	Accepted By: Name	Accepted By: Sign	Not Accepted: Name	Comments
01	21/201	FSB				• Redraw the diagram for Exit Passageway and Exit Corridor. • If a building is office building and the height is less than 24 m, the building can have direct entry to the Exit Staircase. • Add in the area for the Air-well in the illustration. • Delete Item no 6 as its not relevant.
02	01/202	FSB				
03	02/202	FSB				

CLAUSE
Code of Practice for Fire Precautions in Building, 1997
Chapter 2 – Means of Escape
2.2 Determination of Exit Requirements
2.2.13a

AUTHOR:
DATE: 26.03.2001

Objects & Attributes required	IFC Definition
Building	IfcBuilding
Building::NoOfStorey	-
Storey	IfcBuildingStorey
Space	IfcSpcae
Space::Type [ExitStaircase, ExternalExitPassageway, ExternalExitCorridor, Street, ServiceRoad, AirWell,.....]	IfcSpace::Name
Space::IsCoveredWithRoof	-
Space::Width	-
Space::Area	IfcSpace::Area
Ventilation::Type [Natural, Mechanical]	IfcSpace::NaturalVentilation
Wall	IfcWall
Wall::VentilationOpening	IfcOpeningElement

Programming of Code of Practice

Remarks
- What is superficial area?

Code Classification

	IM	CAD	CODE
Easy			
Medium	X		
Hard		X	X
non-Feasi/non-recom			

Usage Scenario

general
- requires space classification for passageways and staircases
- the properties of the air well (beside the footprint geometry) may need to be declared

Scope from tender	Part 2 Annex A – IBP requirements
Within IFC Scope (Y/N/Customise)	Checking Process
Y door, wall → fire resistance supported	additional work to be done to ensure that requirements ""smoke-stop lobby"" and ""external exit passageway"" can be satisfied
C smoke-stop lobby, external exit passageway	
C mechanical ventilation, cross-ventilation, pressurised	same for ventilation

Comments and feedback

Comments and feedback for phase 2
- (i) should check, whether there are ventilation openings within external exit ways or corridors, i.e. if there are (permanent?) openings facing the external space (could be openings, virtual boundaries, balustrades, etc.
  - however, when is it sufficient, e.g. is an opening (let's say 1m²) within a long exit corridor sufficient? Is there any exact measure? How can the system determine it?

Clause relies on clause 2.2.1 – adds additional requirements on top of it
- mainly refines the length and applicability of the exit routes in case of exit staircases

CAD input requirements for phase 2
Clause requires from IFC output
- same as clause 2.2.1

CAD customisation needs
- improved IFC output for code checking view
  - see 2.2.1
- improved CAD functionality
  - see 2.2.1

Figure 2 - A Sample of CORENET ePlanCheck Rule Interpretation Documentation

After CORENET ePlanCheck, there had not been any other major attempt to develop a similar system at the same scale. Only recently, the Korean authority has launched a similar initiative as CORENET (Kim, 2015). Among the activities is rule checking system. From the information available, it follows the production rule approach where rules are broken down into the atomic sentences following a form: **if *<condition>* then *<action>* else *<action>***. It is combined with the object model that encapsulate IFC objects including the geometry and adding methods to the objects as an extension of the rule requirements (J. K. Lee, 2015). This is essentially the same approach as FORNAX (Solihin et al., 2004). It is not obvious at present whether the

knowledge capture involves rule experts. It is also too early to see how this project scale beyond the first phase of rules that usually involves mostly simpler rules (as described as class-1 and a section of class-2 in (Solihin & Eastman, 2015)).

4 SEMANTIC KNOWLEDGE REPRESENTATION OF BUILDING RULES

Recognizing the importance to address the knowledge gap and reducing the information loss, this study looks into a suitable representation to capture semantic knowledge for building rules. The most suitable method appears to be from the field of Knowledge Base (KB) that is a branch of Artificial Intelligence (AI). But even within KB, there are many different approaches including First Order Logic (FOL), Description Logic (DL), and Conceptual Graph (CG). CG was originally proposed by Sowa in 1976 (John F. Sowa, 1976) and further developed in 1984 (J. F. Sowa, 1984). The Conceptual Graph (CG) offers intuitive, easy to read and suitable to capture semantic knowledge representation of the rules (Chein & Mugnier, 2008). CG has a semantic foundation in FOL and the basic form of CG can be mapped 1-to-1 directly to FOL. In this paper, we propose the use of Conceptual Graph (CG). The CG is designed to capture knowledge of the rule into its basic logic structure and the data involved. The aim is to enable effective communication between all users and to identify exact data, relationships between data, and any required functions to encapsulate the complex algorithms involved in solving a rule. The CG if used effectively will allow transparency into otherwise a black-box in the implementation. The white-box approach, as opposed to the black-box approach, allows the use of CG to capture the requirement for derived data and relationships that are vital to rule checking implementation.

Thus, the goals of using the CG are:

- As an expressive tool to capture knowledge of rules in term of their requirements for automation that are easily understood by the rule experts, who are typically not familiar with computer programming.
- Ability to capture data requirements of the building objects and their relationships or interactions with other building objects.
- Direct mapping of the CG concepts into IFC entities, derived entities and extension functions. The mapping is important for both defining MVDs and for software development efforts.
- Ability to break down complex rules into their atomic rules in a systematic and standardized way.

4.1 Conceptual Graph as a Knowledge Representation of Building Rules

With its history in semantic networks, CG defines rectangles to represent concept nodes, ovals represent conceptual relations and diamond shapes represent functions (an extension to CG). The nodes are connected by arcs with an arrowhead pointing to the ellipse, which marks the node as the first argument of the relation. The node with the arrow pointing away from the ellipse marks the last argument (Figure 3).

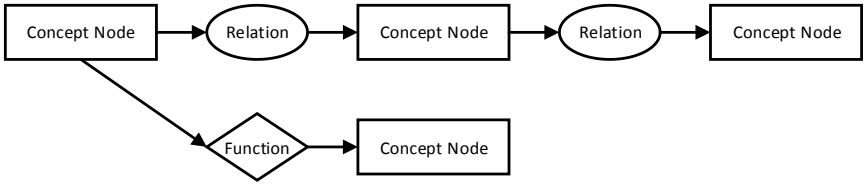


Figure 3 - Basic Definitions of Conceptual Graph

A Concept node typically represents an object, but it can also be extended to represent a whole atomic rule. This is achieved using a concept called coreference that links two different nodes. Coreferent nodes should be able to be merged into just a single node. They are represented by dashed line (Figure 4).

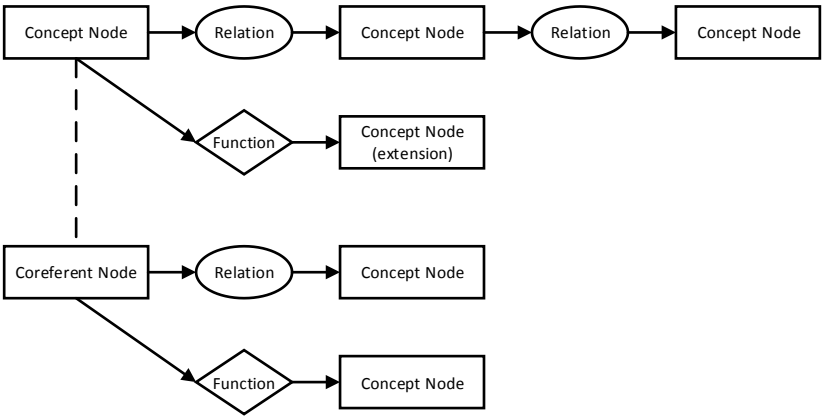


Figure 4 - Coreferent Node

The CG is used to represent the semantic knowledge of the rules. A rule will be represented by a series of connected graphs as above.

4.2 Notational Extension

Due to the complexity of building rules, there is a necessity to add a few notational extensions into the graph for improved readability of the CG. Figure 5 shows such extensions:

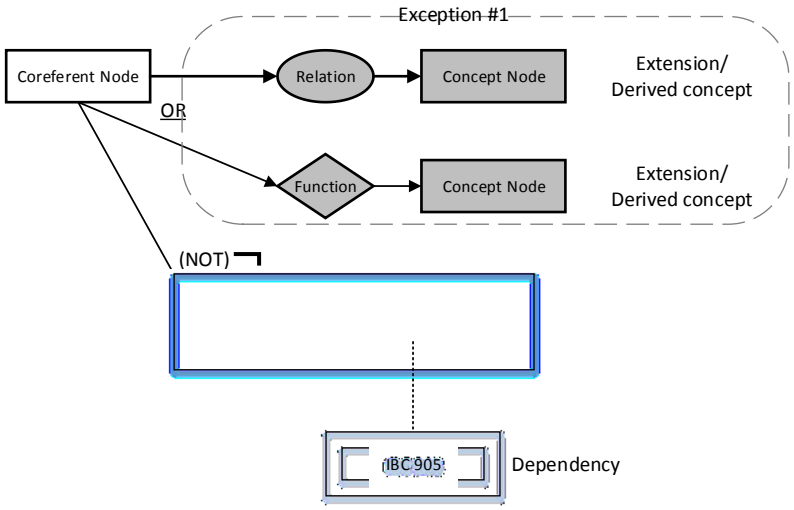


Figure 5 - Notational Extensions for CG

- Nodes that is shaded represent a derived concept or concept that will require additional support during the implementation using computer algorithm. The requirement for a derived concept has been identified to address more complex classes of rules (Solihin & Eastman, 2015). The derived concept usually requires support from a computer algorithm that is applied to the basic building model. FORNAX^{TM} , which was developed for CORENET ePlanCheck, used the same concept (Solihin et al., 2004), and very recently a rule checking effort in the UK took a similar approach (S. Malsane, J. Matthews, S. Lockley, P. E. D. Love, & D. Greenwood, 2015).
- Specific labels: OR and (NOT) ¬ to represent logical disjunction ($\vee$) and negation ($\neg$). By default, if there are more than one link connecting a node, the operation is a logical conjunction ($\wedge$). A thin line box surrounding the negation block is part of the standard CG.
- Thin dashed line with rounded rectangle represents a special block, which could be used to show the exception rule.

- A dotted line connecting a concept node to a double border box indicates a dependency for the concept that is specified in another rule. This is important information that connects a certain concept with a specific rule.

5 TRANSLATION OF RULES INTO THE CONCEPTUAL GRAPH

Translation of rules into CG is not always straightforward sequential mapping because the way the rules are written. Generally, the translation step is done during and after the interpretation activity using the following steps:

1) Identify the main concept that the rule is applicable to. The minimum requirement in this step is to identify a concept without any qualification. Typically the main concept is constrained with a filter or specifications for a specific type. For example:

   - "*Spaces (instance) must be from agreed list*" (Solibri) will identify the concept applicable to this rule as a "Space" instance without any further specification. In First Order Logic (FOL) this rule is represented with $\forall x(Space(x))$
   - "*The underground building shall be equipped throughout with a standpipe system …*" (IBC 2009 405.10) will identify that the concept is applicable to a building, but not just any kind of building. It specifies an underground building. In FOL it is represented with $\exists x(Building(x), Property(x, type: UNDERGROUND))$

   In some cases, the concept may not be explicit. In this case, a level of abduction is needed. For example:

   - "*Model should have components*" (Solibri). In this rule, model refers to the building model as a whole because the rule requires any type of entity can be specified. Since the rule comes from Solibri Model Checker, which provides a template, this rule can only be operational once user assigns what entity type(s) that the rule should apply to.

2) Identify atomic sub-rule(s). A rule, especially in building codes, often specifies more than one sub-rules that are independent and are operating on the same entity. In this case, the sub-rules will be defined as separate rules under the same heading. For example:

   - "*Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 inches (178 mm). Doors in any position shall not reduce the required width by more than one-half. Other nonstructural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 11/2 inches (38 mm) on each side.*" (ICC, 2009).

   This rule is applicable to doors that open to the egress path. There are three sub-rules, one deals with the space occupied by the door at the fully open position, the second one deals with the reduction of egress width due to the door opening, and the third one specifies maximum projection of the trim and decorative features of the door into the egress space.

3) Identify atomic constraint(s). The general structure of a building rule is a specification of the main building entity with its details, followed by one or many constraints. The constraints are not restricted to the main entity, but can also apply to other entities that are related to the main entity or even to an entity within the constraint, i.e. constraint within constraint. This increases the complexity of the rule. For example:

   - "*All patient rooms must be visible from the nurse station*"

   This rule has a constraint on the main entity, the nurse station. The constraint specifies that the patient rooms must be within the line of sight from the nurse station.

4) Define the appropriate CG of the rule by connecting the concepts using relations and functions until the consistent semantic is clearly self-describing.

5.1 Applying the Semantic Representation CG to Building Rules

In this section, several rules are selected to represent the ranges of rules that are applicable to buildings. Several rules are selected from different classifications of rules as defined in (Solihin & Eastman, 2015), mainly for class-1 to class-3. Class-4 does not create new semantics or complexity, but it introduces requirements in term of the algorithmic solution to present a "proof of solution". Therefore, it does require additional semantics than the other three classes.

5.1.1 Class-1 rule (rules that require a single or small number of explicit data) example

"Spaces must be from agreed list" (Solibri)

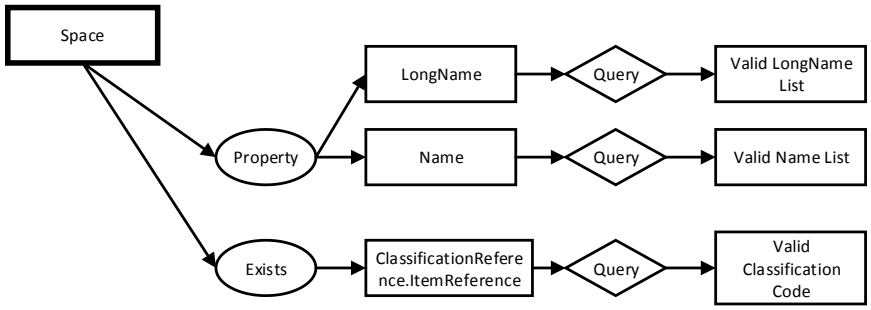


Figure 6 - Class-1 rule example

In FOL, the above rule (Figure 6) can be expressed as:

$$\forall a\big(Space(a)\big) \wedge \exists a((Space(a) \wedge Query(a, Name)) \wedge \big(Space(a) \wedge Query(a, LongName)\big) \\ \wedge \exists a((Space(a) \wedge Query(a, IfcClassificationReference.ItemReference))$$

This rule checks the existence of properties *Name* and *LongName*, and checks *IfcClassificationReference.ItemReference* using a simple query function that checks for the existence of a property or a classification. *Name* and *LongName* properties are existing properties in IFC schema and *IfcClassificationReference* is the IFC entity that is used to assign classification item to an entity, which is expected in this case for an *IfcSpace*.

5.1.2 Class-2 rule (Rules that require simple derived Attribute Values) example

(42) 3.2.2 Design Criteria (Singapore, 2013)

f) The discharge pipe shall not be located in places where it can cause health and safety hazards such as locating the discharge pipe above any portable water storage tank and electrical transformer/ switchgear.

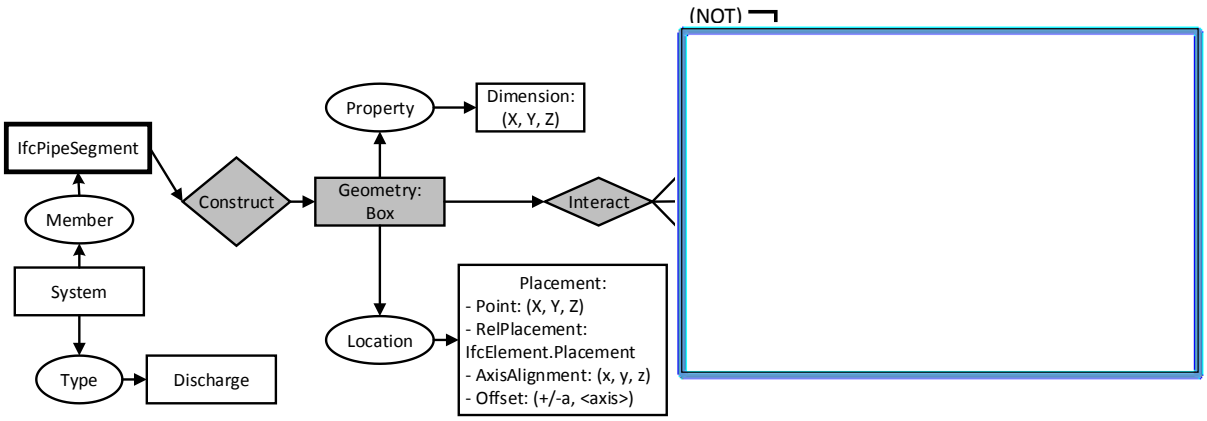


Figure 7 - Class-2 Rule Example

Representation in FOL for the above CG (Figure 7):

$$\exists b \forall a(\forall g(IfcPipeSegment(a) \wedge System(b) \wedge Discharge(g) \wedge Member(b,a) \wedge Type(b,g)) \wedge (\forall c \forall p \forall d(Box(c) \wedge Placement(p) \wedge Dimension(d) \wedge Location(c,p) \wedge Property(c,p)) \wedge Construct(a,c)) \wedge \neg(\exists w \exists t \exists m(Interact(c,(\forall u(IfcTank(w) \wedge Usage(u,POTABLE) \wedge Property(w,u)) \vee IfcTransformer(t) \vee IfcSwitchingDevice(m))))))$$

This expression requires an extension function to construct a transient Box geometry based on a specified location and dimension. This Box is used to evaluate the existence of any type of object within the Box that is a type of a potable water tank, a transformer, or a switching device. Their existence within the Box is not allowed.

As shown in Figure 7, in this class-2 rule, we start to see the need for extensions to generate a new concept. Three such extensions are required in here: a simple box geometry and two functions to construct the box and to perform a spatial operation to find the specific object types that interact with the constructed box that is placed below the discharge pipe. Since this rule comes from CORENET ePlanCheck, it is interesting to see how the CG representation compares to the original interpretation document (Figure 8). From this comparison, CG clearly is more compact, precise and self-describing compared to the original specifications used in CORENET ePlanCheck.

This example also presents an interesting possibility that the CG specifications can also be done in a different way. During the actual implementation of this rule, one will notice that applying the rule to all instances of discharge pipes may require quite significant computing efforts due to the potentially large number of the pipes. However, the number of instances of the potable water tank, switchgear, or transformer, should be much less in the model. Therefore, it is more practical to invert the order of the rule definition. With this, the specific equipment becomes the main objects to check and the pipes become the objects that defined the constraint criteria. The logic of checking does not change much, except that the transient bounding box now should be constructed upward instead of original specifications, which is downward. The inverted version of the CG is shown in Figure 9.

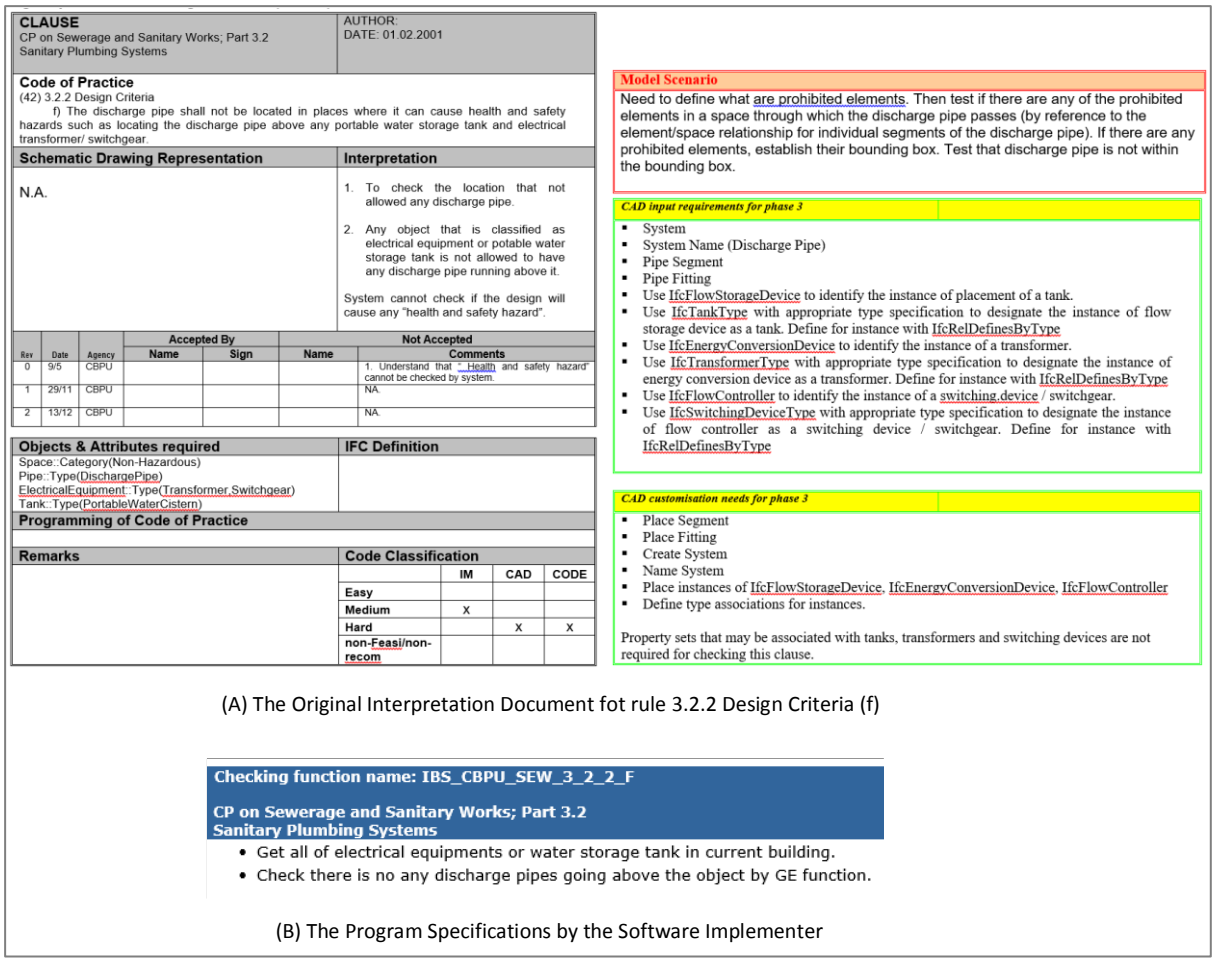


Figure 8 - The Original CORENET ePlanCheck Interpretation Document and Program Specifications for Rule 3.2.2 (f)

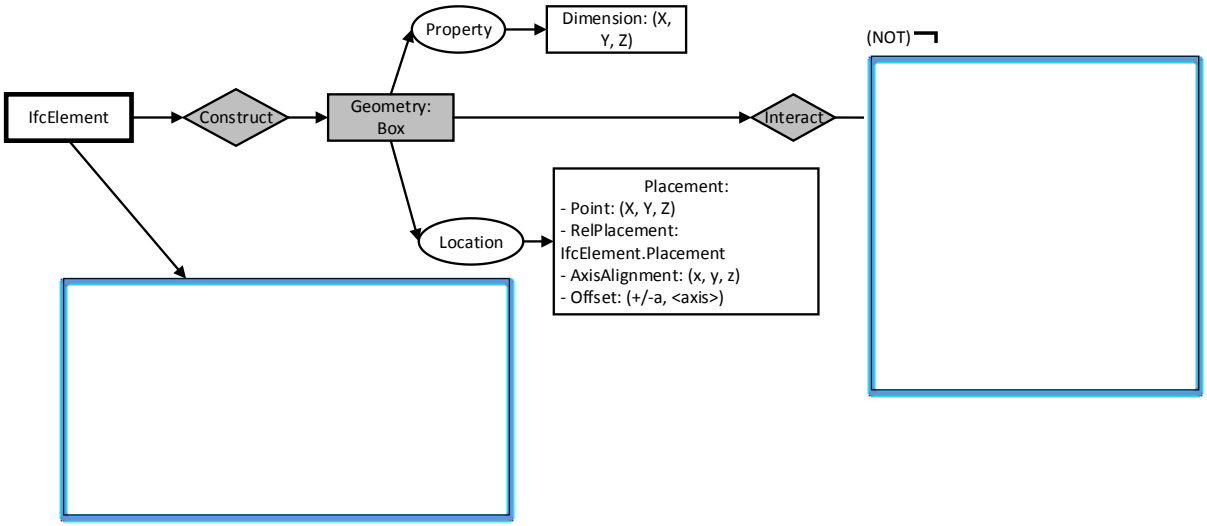


Figure 9 - The Inverted Version of CG Specification for Rule 3.2.2 (f) as Shown in Figure 7

5.1.3 Class-3 rule (Rules that require extended data structure) example

IBC 1005.2 Door encroachment. Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 inches (178 mm) ❍,1. *Doors in any position shall not reduce the required width by more than one-half* ❍,2. *Other non-structural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 1-1/2 inches (38 mm) on each side* ❍,3. *(ICC, 2009)*

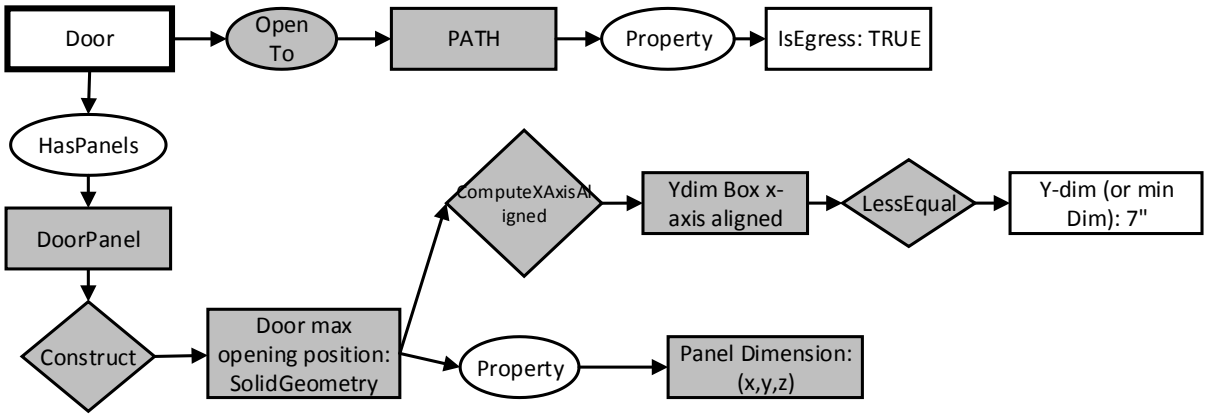


Figure 10 - Class-3 Rule Example – Sub-rule #1

In FOL, the above graph for sub-rule #1 can be expressed as:

$$\forall d \exists p((Door(d) \wedge OpenTo(d,p) \wedge PATH(p) \wedge Property(p, IsEgress: TRUE)) \wedge \forall q((DoorPanel(q) \wedge HasPanels(d,q) \wedge Dimension(r) \wedge Property(q, r: (x,y,z))) \wedge \forall b \forall y(Box(b) \wedge YDim(y) \wedge Construct(q,b) \wedge ComputeXAxisAligned(b,y) \wedge LessEqual(y, minYDim: 7")))$$

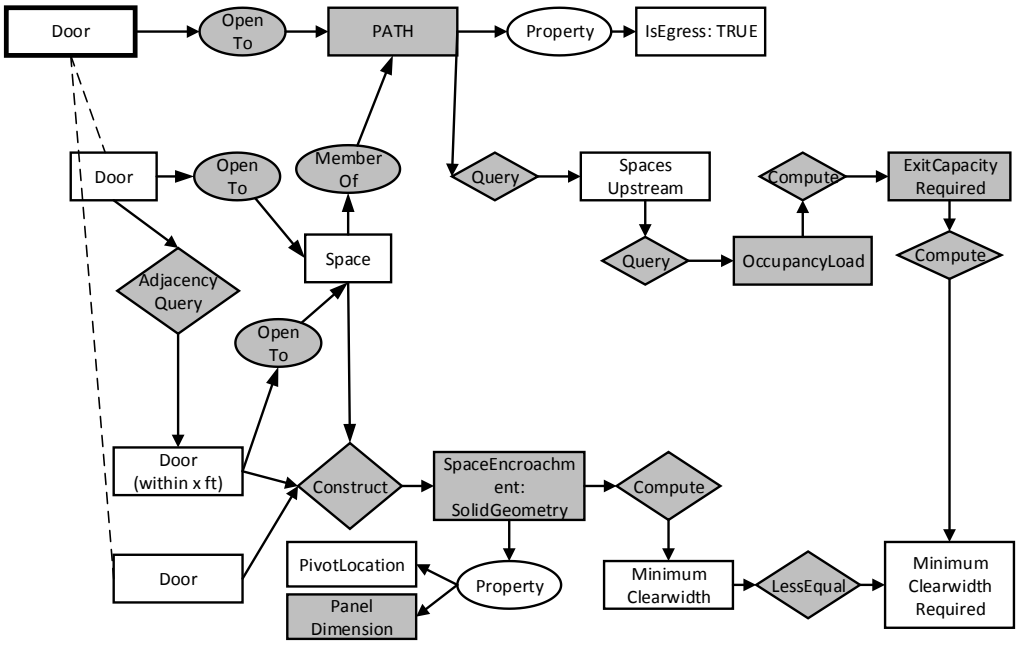


Figure 11 - Class-3 Rule Example – Sub-rule #2

Sub-rule #2 can be expressed in FOL as:

$$\forall d \exists p \Big(\big(Door(d) \wedge OpenTo(d,p) \wedge PATH(p) \wedge Property(p, IsEgress: TRUE) \big) \\ \wedge \exists y \exists x (Space(x) \wedge MemberOf(p,x) \wedge OpenTo(d,x) \\ \wedge DoorList(y) \wedge AdjancencyQuery(d,y,Distance) \wedge OpenTo(y,x)) \\ \wedge \forall s \forall o \forall w \big(Space(s) \wedge QueryUpstream(p,s) \\ \wedge OccupancyLoad(o) \wedge ExitCapacity(c) \wedge Query(s,o) \wedge ClearWidth(w) \\ \wedge Compute(o,c,w) \big) \\ \wedge \forall l \forall m \forall e \forall v \big(Pivot(l) \wedge Panel(m) \wedge Property((d,y),(l,m)) \\ \wedge SpaceEncroachment(e) \wedge Construct(x,(d,y),e) \wedge Property(e,(l,m)) \\ \wedge MinClearwidth(v) \wedge Compute(e,v) \wedge LessEqual(v,w)) \Big)$$

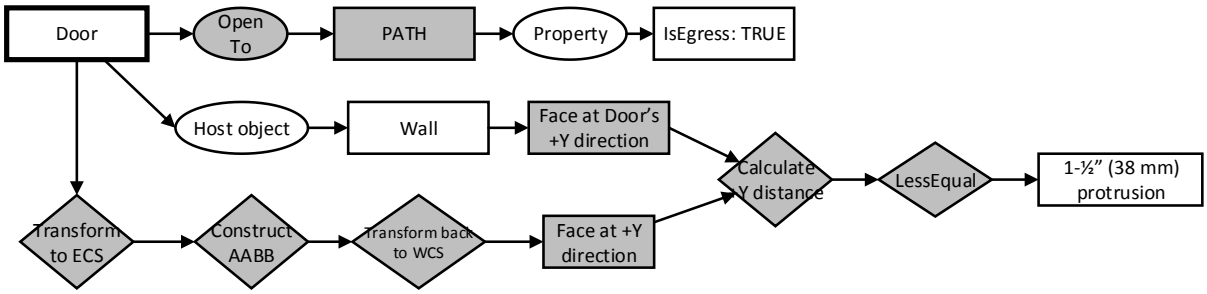


Figure 12 - Class-3 Rule Example – Sub-rule #3

Class-3 rules require extensive extensions both to the building model in the form of derived data as well as functions for the purpose of computation, which includes geometry operations. The above rule highlights the nature of complexity besides the needs for an extension, but also nested and branching conditions of the sentence that can occur in any entity within the statement.

Sub-rule #2 in this example is an entirely independent rule from the sub-rule #1, except that it applies to the same main entity. Here, it is perfectly ok to separate the sub-rule as a new independent rule. In other cases, the sub-rule may serve as an exception to a nested rule inside the main rule. In this case, the use of coreferent concept will become handy.

Sub-rule #3 requires computation of the amount of space occupied by door protrusion into the space. It can be computed using its Optimized Bounding Box (OBB) and compared to the face of the space that it is facing.

6 THE GRANULARITY OF THE KNOWLEDGE CAPTURE

For the purpose of capturing knowledge for rule requirements and communication, the level of granularity of the information is important. Too coarse of the granularity risks the requirements to be ambiguous, but too fine of the granularity will also increase complexity and makes the specifications unreadable. One has to strike a balance that the essential details of the knowledge should be captured but some details may be left to the implementation. However, there should be no ambiguity of what the item is all about. To illustrate this, we use an example from the best practice used in a modern hospital design in the USA:

> *All patient rooms shall be visible from the nurse station*.

At the first glance, the above rule may be captured simply as followed (Figure 13).

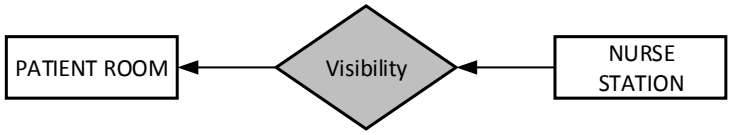


Figure 13 - The highest level of CG that captures the patient room visibility rule

The patient room and nurse station are both represented as spaces and, therefore, needs to be defined using the relationship type (Figure 14). In this CG, the function "Visibility" does not have any details. Thus, this knowledge is too coarse of the granularity and it is equal to a black-box.

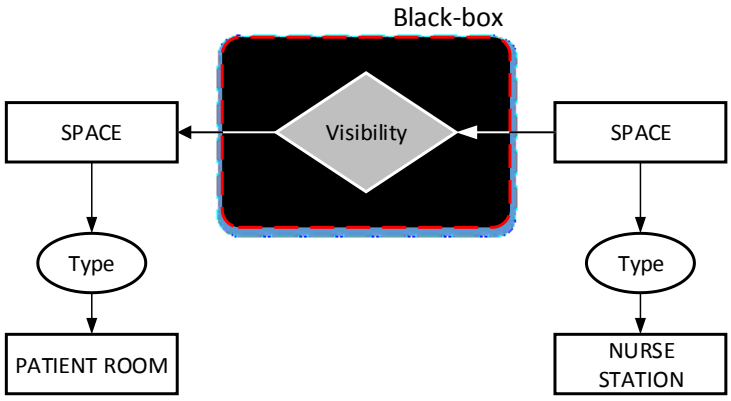


Figure 14 - The visibility rule as a black-box

One possibility to refine the granularity is to limit the visibility for the patient room and the nurse station only to those on the same floor (Figure 15). However even with this refinement, the visibility is still a black-box. To further the refinement, the visibility needs to be further defined with additional input from the rule experts. One of the approaches for the visibility is using a line of sight from the nurse station to the opening of the patient room that allows visibility such as a door, opening or a glass window. The access to the patient room is typically from the nurse station that shares the same corridor space using the boundary information between spaces and element connecting the spaces such as the door, opening or window.

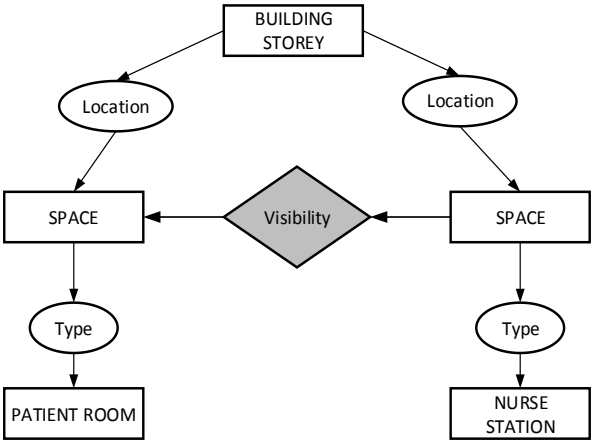


Figure 15 - a finer granularity of the visibility rule

Extending the above diagram to incorporate this additional specifications results in Figure 17. In this diagram, there is also another checking function introduced (checking function #2). The purpose of this checking function is to compute the actually visible portion of the patient room by computing the volume intersection of the patient room and a frustum created by connecting the starting point at the nurse station location and the ray extended plane of the door, opening or window. By the percentage of this volume intersection, the estimated real visibility can be computed and compared. Figure 16 describes this checking requirements visually.

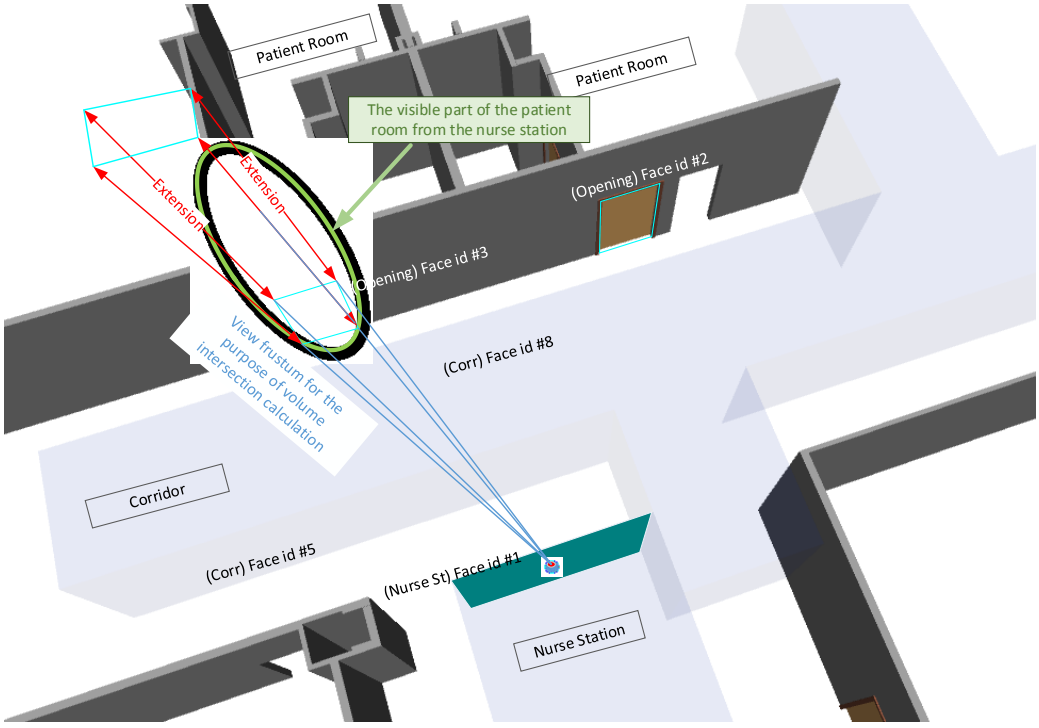


Figure 16 - Visual description of constructing the line of sight and the view frustum

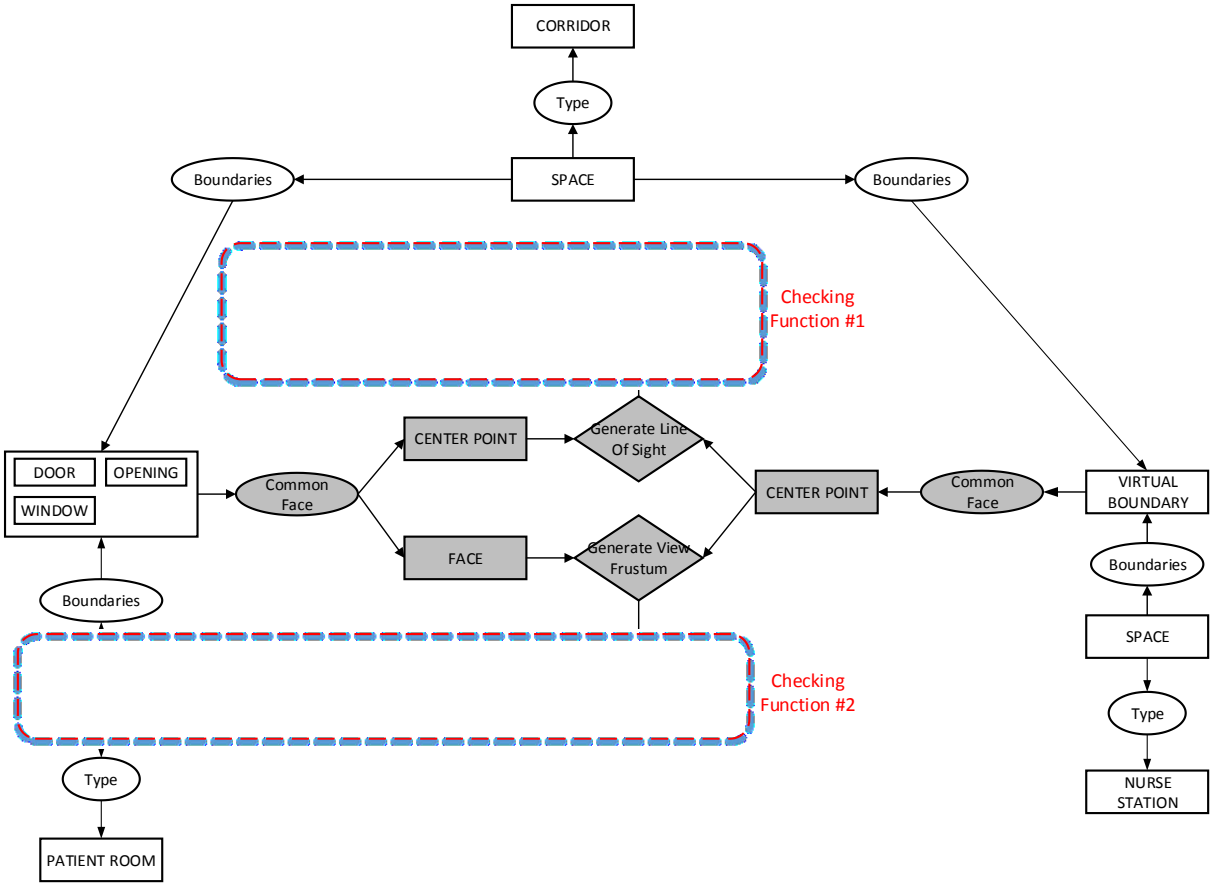


Figure 17 - An even finer granularity of the visibility rule, the visibility is no longer a black-box

These rule requirements can be extended further by adding the possibility to measure the visibility from a point in the nurse station using a position from the desk (furniture) within the nurse station location. Figure 18 represents complete rule requirements in a fine granularity that reflect transparent checking details without creating too highly complicated diagram.

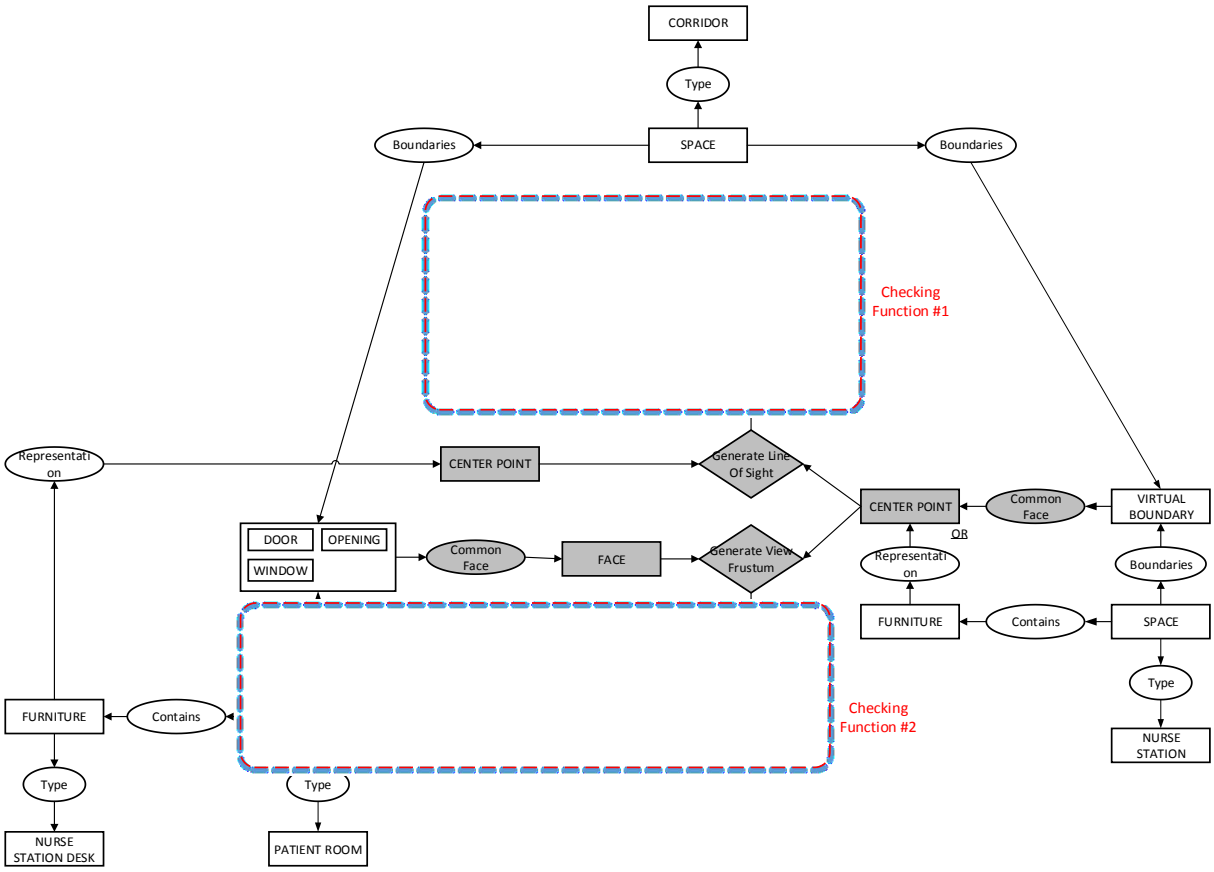


Figure 18 - Another variation of the visibility rule that allows more option for defining the line of sight

7 MAPPING THE CG TO THE MVD

With the well-defined CG, it is possible to create a direct mapping of CG concepts and relations into an IFC MVD as well as to a UML diagram for software development (Figure 19). In the mapping to an MVD, IFC entities represented by Concept and Relation can be directly mapped to the IFC MVD, which includes relevant details such as Types and Properties. Each of the rules can be defined as one exchange requirement within the MVD. In practice, it may be practical to define only one or just a few specific MVDs, in order to consolidate the MVD requirements since they often overlap with other rules.

Extended Concepts and Functions do not have equivalent mappings to an MVD and are only applicable for mapping to the software design, represented using a UML diagram in this example. Figure 19 shows an example of mapping from CG to an MVD and to a UML diagram. This diagram uses the example for a class-2 rule given earlier in section 5.1.2 (Figure 7).

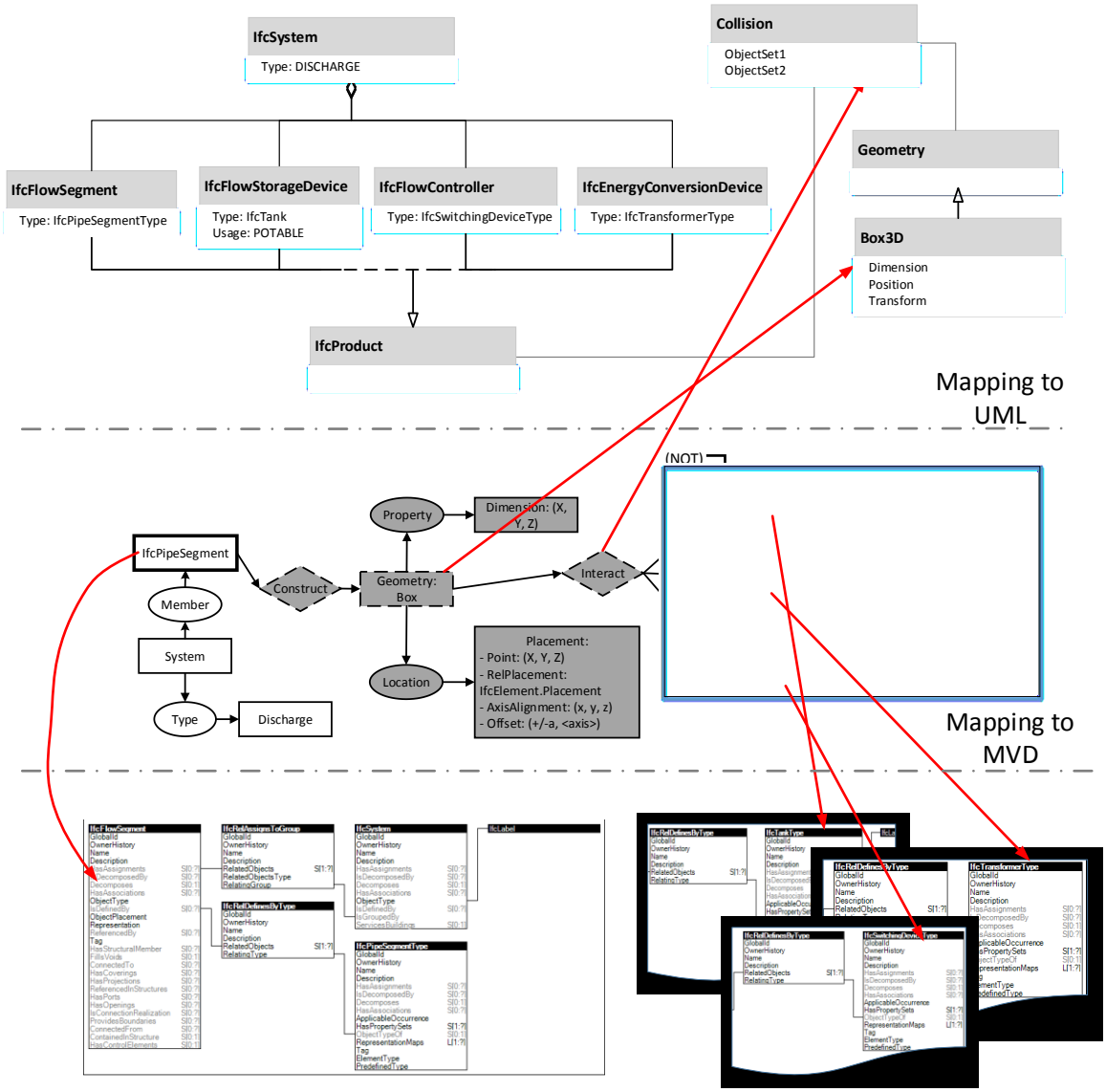


Figure 19 - Mapping CG into an MVD and UML

8 VALIDATING THE CG AGAINST THE ACTUAL IMPLEMENTATION

The CG as the tool to capture knowledge contains information where the data should come from and what information is needed to perform the checking. It does not capture the detailed algorithm to perform the checking. It is important therefore to be able to validate how effective is the knowledge capture using CG as compared the actual implementation. Since one of the rules used in the examples is taken from the scope of the actual implementation of CORENET ePlanCheck, comparing the codes written for the rules in CORENET ePlanCheck that was completed in 2005 will give an indication of the effectiveness of the knowledge capture. The rule that comes within the CORENET ePlanCheck scope is rule 3.2.2 (f) that is described in details in section 5.1.2. It is also used to describe mapping the CG to MVD and UML in section 7. Figure 20 shows the snapshot of the CORENET ePlanCheck implementation for the rule (written in C++ programming language) with the relevant boxes that highlight corresponding section in the CG.

Another validation that can be done is against a rule language developed as part of a Ph.D. thesis (Wawan, 2016), named BIMRL (BIM Rule Language). It is a domain specific language designed specifically for automating BIM rule checking. It deals with defining a simplified schema in a relational database to represent a read-only building model including its geometry. It also supports spatial operations for rule execution. In another word, BIMRL is a complete environment for data, rule definition, and rule execution. As part of the validation

tests in the thesis, rule 3.2.2(f) is also used as one of the proof-of-concept test cases. In BIMRL, the optimized approach of using the inverted rule described in the section 5.1.2. The BIMRL description of the rule maps very closely to the CG representation as shown in Figure 21.

The second validation example is the hospital design rule described in details in section 6. Since this rule is not in the scope of CORENET ePlanCheck, it can only be validated against BIMRL. Figure 22 shows how the knowledge capture is represented very closely in the BIMRL script that implements the rule, without the use of black-box specifications.

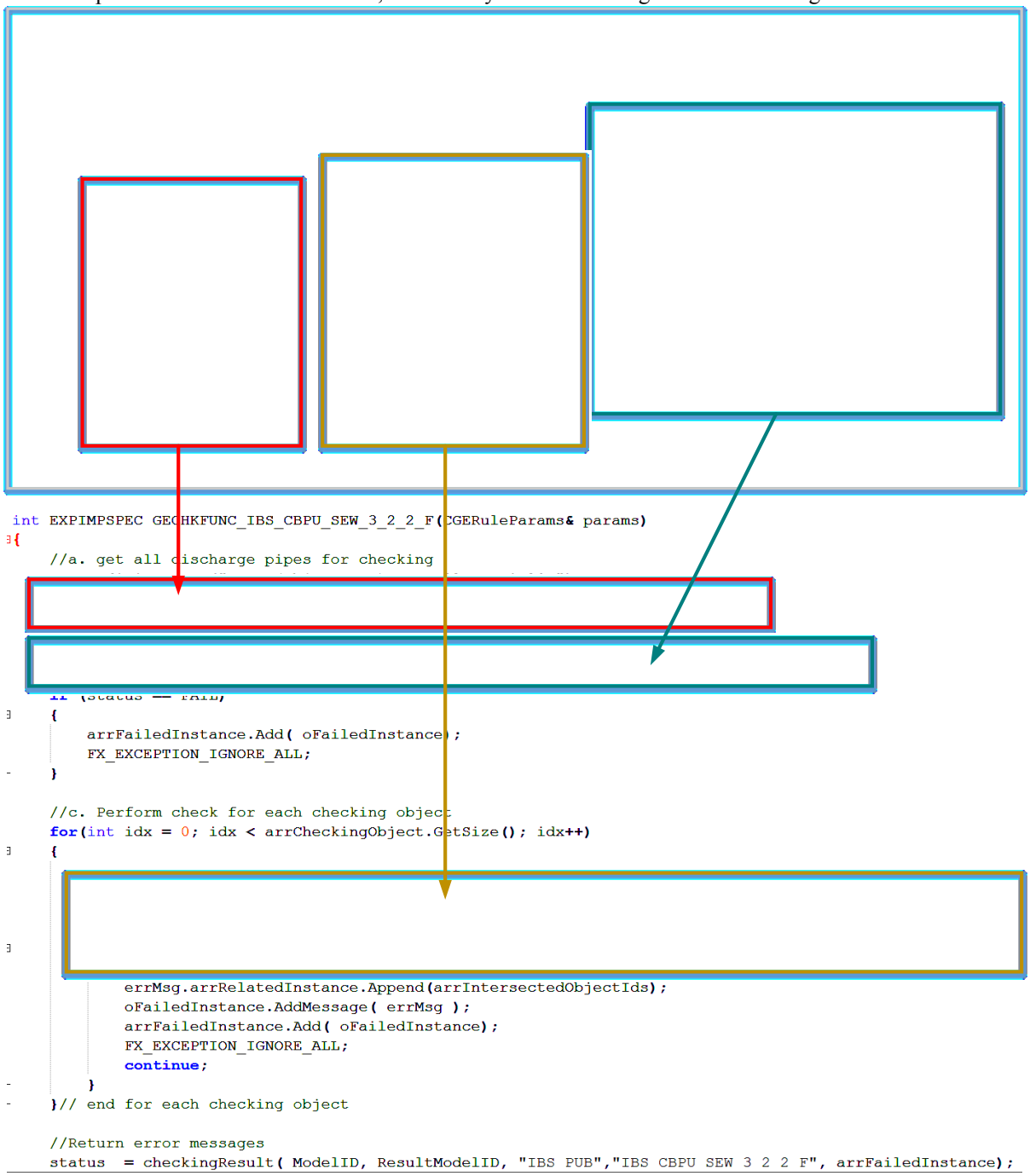
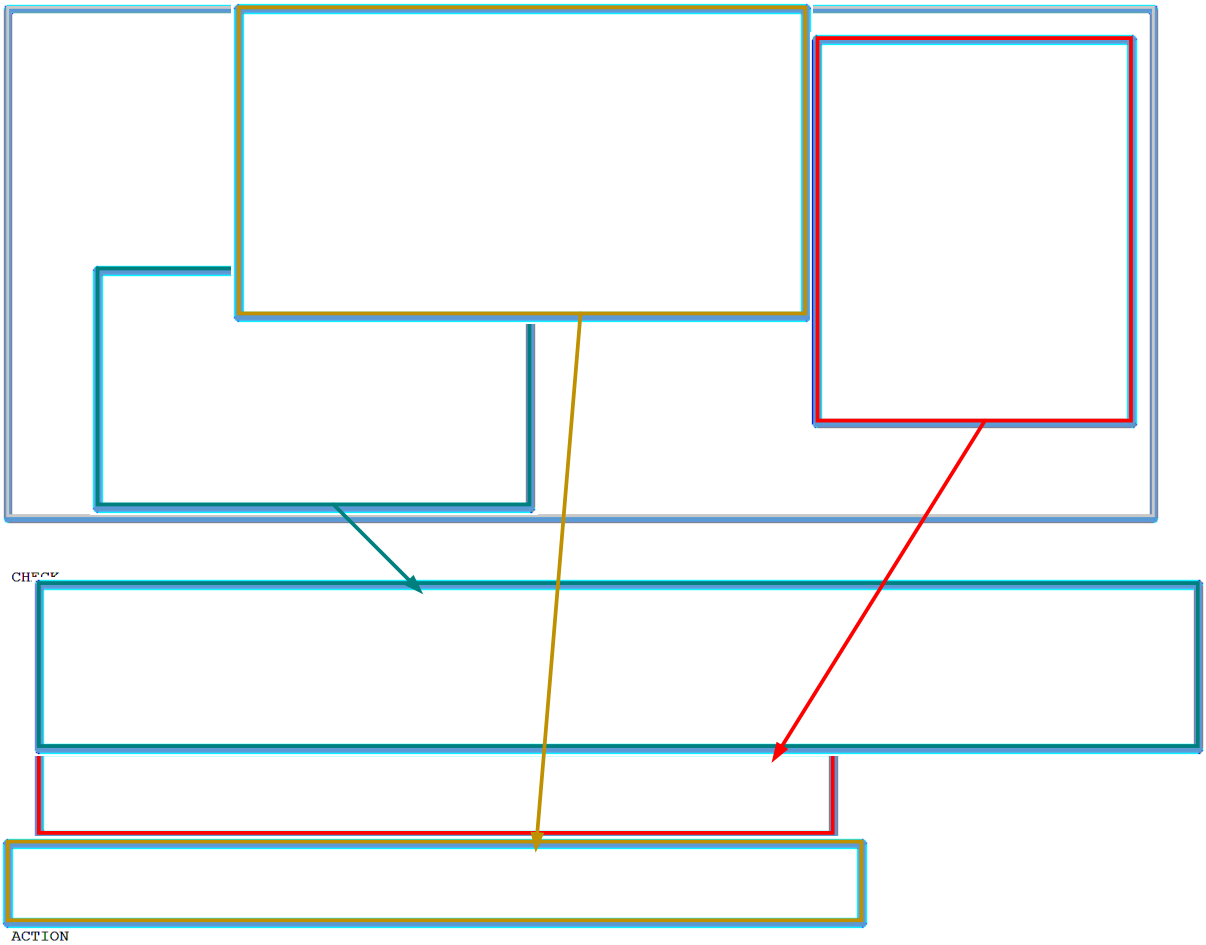


Figure 20 - The CG for Rule 3.2.2(f) compared to the actual code implementation in CORENET ePlanCheck



```
ACTION
    WHEN ?clash = 0
    { PRINT RESULT SAVE INTO TABLE contamination DRAW COLOR CYAN TRANSPARENCY 0.5 WITH TRANSPARENT BACKGROUND
        (IfcDistributionElement, IfcSlab, IfcCovering)
        highlight (OBJEID, OUTPUTDETAILS) COLOR RED TRANSPARENCY 0 SAVE INTO X3D 'c:\temp\NoContamination.x3d' };
    WHEN ?clash > 0
    { PRINT RESULT SAVE INTO TABLE contamination APPEND DRAW COLOR MAGENTA TRANSPARENCY 0.5 WITH TRANSPARENT BACKGROUND
        (IfcDistributionElement, IfcSlab, IfcCovering)
        highlight (OBJEID, OUTPUTDETAILS) COLOR RED TRANSPARENCY 0 SAVE INTO X3D 'c:\temp\contamination.x3d' };
```

Figure 21 - The CG for Rule 3.2.2(f) (the inverted version) as compared to the BIMRL script that implements the rule

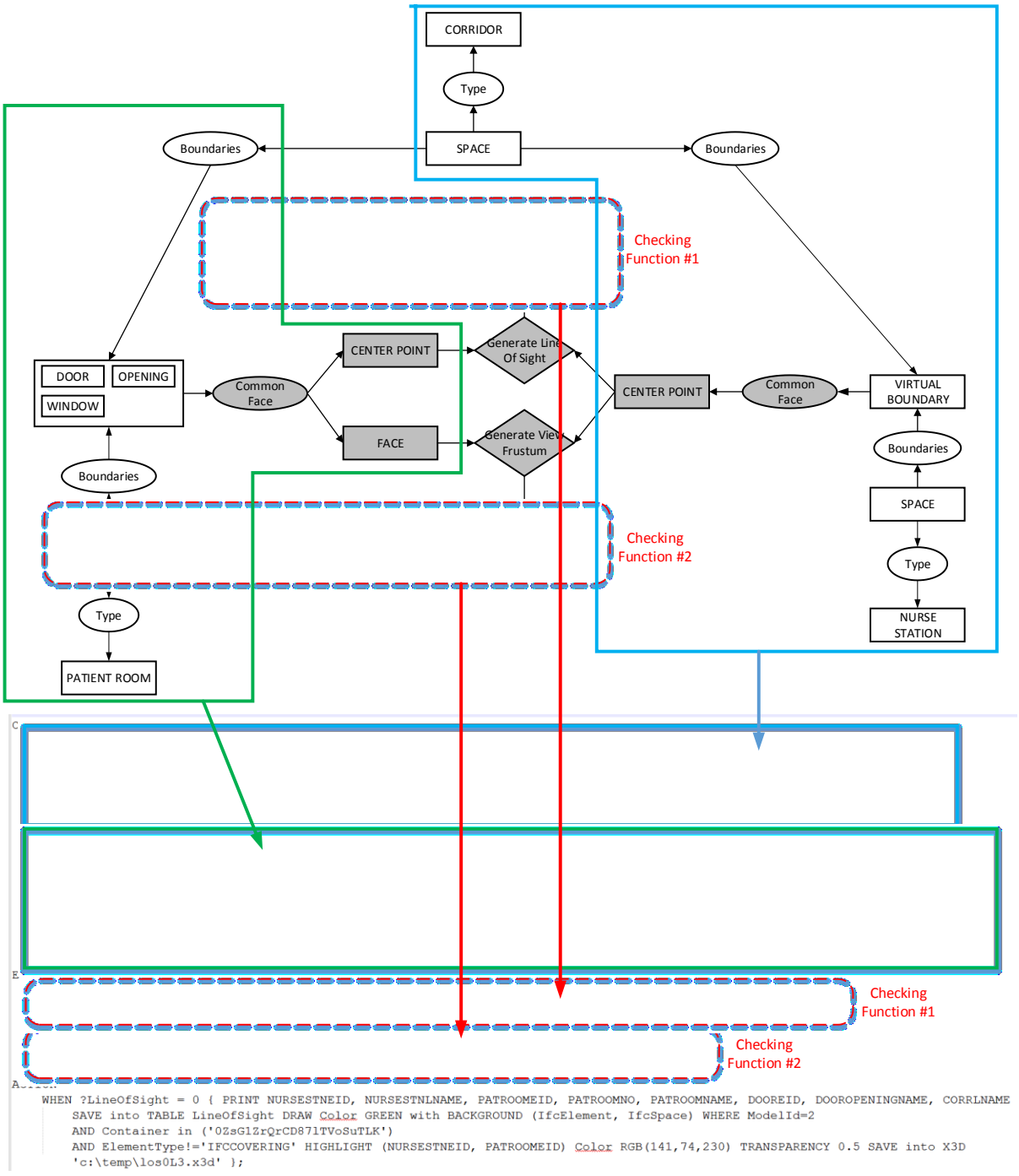


Figure 22 - Another Validation for the CG for a hospital rule against the BIMRL script

9 DISCUSSIONS ON THE GENERAL CHARACTERISTICS OF THE RULE STRUCTURE

From the exercise of describing the rules into CG, building rules may range from a simple to a complex structure. The complexity is often man-made because of the way the rule is written in many iterations of edits by different people over a period of time, and also the rules that are by nature complex due to the requirements they present. The former complex structure may be simplified by breaking the rule into smaller sub-rules. It is especially easy if the sub-rules are just a combination of various logic applicable to the same entity. An example from IBC 1008.1.2 (Figure 24) shows that several of the exceptions are applicable to the same entity for various

building types or entity types. Note that the exceptions to a rule are generally rules themselves. The general logic structure of the rules, thus, can be summarized as follow:

1. The rules generally take the form of:
    a. An entity in focus, i.e. an entity where checking is to be done. For example Space
    b. Further description of the entity, usually condition(s) that will narrow down the entity. For example Circulation space, or Egress Path
    c. A specific entity that is defined by another entity. For example Vestibule space where an exit discharges into (see IBC 1027.1 rule Figure 29 - Figure 31) is what Description Logic (DL) defines to be Noun Phrase.
2. Rules in the building codes are often applicable only to a certain type of building type or occupancy type. This is an important feature that will help users to quickly identify what rules to check for the specific type of developments. Currently, this information is not captured in the examples given in this paper. They are generally independent of the logic structure and only gives further restrictions mainly for search purposes. This condition can be added to the beginning of the CG with the building as a constraint entity and is possibly further restricted by its property. In the case of mixed use buildings, a relevant constraint that is usually applied to separate building stories can be assigned to the main object. Since CG allows assignment of constraint to any concept node, rule definition can be tailored relatively easily to filter applicability of the rule based on the constraint defined to the specific concept node.
3. The sentence can have sub-sentences or phrases starting at any entity within the sentence. The phrase may take any of the following forms:
    a. A phrase added to the sentence
    b. A phrase added to a phrase, creating nested constraint
    c. Branch, when a phrase is added in parallel to another phrase. As part of b, a branch can have another branch creating a tree-like structure, even though usually it is not that deep. Coreference usually is represented by a branch.
    d. Merge, when two branches meet again at some point in the sentence
4. Function or Verb forms a very important part of the sentence especially at the higher level of rule classification. Functions are general to be expected to be supported by computer algorithms in the form of a library in a rule checking system. The capability of such a rule checking system will be a direct proportion of the number of unique verbs in its vocabulary.
5. Higher level derived data is an integral part of rule structure. Many types of information required in the rules are not usually in the model and many are simply impractical to be expected to be explicitly included in the model. For example, a rule that requires a check of the distance between doors connecting to circulation space from a single room. The distance information cannot be anticipated beforehand and it depends on the model data. Other types of information may involve information that is difficult to get manually and will be easier done by a computer. For example, a graph that represents space connectivity or access to the entire building.
6. Exceptions. Many building codes contain one or many exceptions. From a closer look an exception may take several forms:
    a. As an additional constraint to the main rule. It may be a negation or a more specific condition where the rule can be ignored or another rule is supposed to be adhered to.
    b. As a form of sub-rule
    c. As an independent rule on the same entity
    d. As a phrase that can be inserted at certain points of the sentence, often as a negation.
7. Rule dependency. Rule dependency is a rather ambiguous definition often found in the rule. It is ambiguous because the dependency is often very loose. For example, if a rule has an exception that in a certain condition when there is an existence of a specific entity (e.g. a railing as a form of protection),

the entity must comply with rules under a certain section. Unfortunately, not all requirements in those sections related to the entity are applicable for this condition and in some cases there are other exceptions that lead to a circular reference. Generally, the strategy to deal with rule dependency is one of the following:

a. Ignore it since it will be captured when the rule is handled by the referred entity.

b. Create a dependency list for a rule checking system to automatically include dependent rules during execution. It may also need to order the execution accordingly. Care needs to be taken in this case not to create a situation where it leads to a very large dependency tree due to unrelated dependency within the dependent rules. It also needs to ensure that it knows how to handle a circular reference.

c. Define it as a sub-rule if it is distinct enough. This may be required in a very small number of rules, especially when there is tighter dependency such as one result affecting the other.

Due to the generally complex form of rules, especially building codes, it is generally a good idea to try to separate one large rule into independent sub-rules. It will make the rule more discernable and hence easier to maintain. This is beneficial even if there are small redundancies between sub-rules due to some minor variations in the sub-rule. Many exceptions as described above can be either separated as an independent rule, as a sub-rule or inserted into the main rule.

10 CONCLUSIONS

In this paper, we have presented the case to treat interpretation of the rules as knowledge representation. It is critical that such knowledge can be captured and retained throughout the entire process of rule checking implementation with minimum loss of information (Figure 1). The use of Conceptual Graph as a tool to capture the building rule requirements and their general checking logic looks very promising. It is effective in capturing both data requirements and the higher level checking logic in an intuitive format, which can be understood by typical rule experts with a little training. The exercise to capture the rule expert knowledge into CG also provides a template for analysis and breaks down a complex rule into atomic rules and constraints. In our research work, the method has been applied to more rules of higher level complexity and it is found to be able to describe the rule requirements and logic successfully. The CG also has been shown to improve clarity and precision of the rule requirements compared to the traditional documentation such as used in CORENET ePlanCheck project.

The proposed CG allows a direct mapping to an MVD for model exchange requirement relevant to the rule it describes, and a mapping to software class design such as a UML diagram. The current mapping process is still done manually, however with well-defined CG, an automated mapping process from the CG directly into an MVD and software classes is possible. Similar work to capture requirements using CG-like representation into UML has been done (Jaramillo, Gelbukh, & Isaza, 2006). While it is not specific to building rules that involve the additional complexity of derived entities, properties, and functions, it may give an idea of what can be done in the same direction for building rules. The potential time saved and knowledge retention throughout the whole development process is extremely significant.

REFERENCES

Beach, T., Kasim, T., Li, H., Nisbet, N., & Rezgui, Y. (2013). Towards Automated Compliance Checking in the Construction Industry. In H. Decker, L. Lhotská, S. Link, J. Basl, & A. Tjoa (Eds.), *Database and Expert Systems Applications* (Vol. 8055, pp. 366-380): Springer Berlin Heidelberg.

Bouzidi, K. R., Fies, B., Faron-Zucker, C., Zarli, A., & Thanh, N. L. (2012). Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry. *Future Internet, 4*(3), 22. doi: 10.3390/fi4030830

Chein, M., & Mugnier, M.-L. (2008). *Graph-based knowledge representation: Computational foundations of conceptual graphs*: Springer.

Eastman, C., Jeong, Y., Sacks, R., & Kaner, I. (2010). Exchange Model and Exchange Object Concepts for Implementation of National BIM Standards. *Journal of Computing in Civil Engineering, 24*(1), 25-34. doi: doi:10.1061/(ASCE)0887-3801(2010)24:1(25)

Eastman, C., Lee, J.-m., Jeong, Y.-s., & Lee, J.-k. (2009). Automatic rule-based checking of building designs. *Automation in Construction, 18*(8), 1011-1033.

Eastman, C., & Sacks, R. (2010). Introducing a new methodology to develop the information delivery manual for AEC projects. *CIB W78*, 16-18.

Fenves, S. J. (1966). Tabular decision logic for structural design. *Proceedings of the American Society of Civil Engineers, 92*, 473-490.

Fiatech. (2012). Fiatech Autocode phase 1 report.

GSA. Courts Design Guide *http://www.gsa.gov/graphics/pbs/Courts_Design_Guide_07.pdf*.

Hjelseth, E., & Nisbet, N. (2010). *Exploring semantic based model checking.* Paper presented at the Proceedings of the 2010 27th CIB W78 International Conference.

Hjelseth, E., & Nisbet, N. (2011). *Capturing normative constraints by use of the semantic mark-up (RASE) methodology*. Paper presented at the CIB W78 2011 28th International Conference-Applications of IT in the AEC Industry.

ICC. (2009). IBC 2009 *http://shop.iccsafe.org/codes/2009-international-codes/2009-international-building-code-tab-combo.html*.

ISO. (2013). ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51622.

Jaramillo, C. M. Z., Gelbukh, A., & Isaza, F. A. (2006). Pre-conceptual schema: A conceptual-graph-like knowledge representation for requirements elicitation *MICAI 2006: Advances in Artificial Intelligence* (pp. 27-37): Springer.

Khemlani, L. (2005). CORENET e-PlanCheck: Singapore's Automated Code Checking System. *http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html*. Retrieved 03/01/2014, 2014

Kim, I. (2015). Automated Building Code Checking Related Activities in Korea [Presentation]. online at http://iug.buildingsmart.org/resources/London/Regulatory%20Room/korea_automated-building-code-checking-related-activities-in-korea_khu.

Lee, J. K. (2011). *Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program.* (published Ph.D. Dissertation), Georgia Institute of Technology.

Lee, J. K. (2015). Development of the database and logic system for translating sentences into computer executable code (Vol. Regulatory Room). online at http://iug.buildingsmart.org/resources/London/Regulatory%20Room/korea_development-of-the-database-logic-system-for-translating-sentences-into-computer-executable-code_hyu: buildingSMART.

Lee, J. M. (2010). *Automated checking of building requirements on circulation over a range of design phases.* (published Ph.D. Dissertation), Georgia Institute of Technology.

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

Malsane, S., Matthews, J., Lockley, S., Love, P. E. D., & Greenwood, D. (2015). Development of an object model for automated compliance checking. *Automation in Construction, 49, Part A*(0), 51-58. doi: http://dx.doi.org/10.1016/j.autcon.2014.10.004

novaCITYNETS. (2000). novaSPRINT awarded CORENET Integrated Plan Checking System. *http://www.novacitynets.com/news.htm*. Retrieved 03/01/2014, 2014

OGC. (2012). GeoSPARQL - A Geographic Query Language for RDF Data. online at https://portal.opengeospatial.org/files/?artifact_id=47664: Open GIS Consortium.

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

Preidel, C., & Borrmann, A. (2015). *Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling*. Paper presented at the International Symposium on Automation and Robotics in Construction and Mining (ISARC 2015), Oulu, Finnland.

Salama, D. M., & El-Gohary, N. M. (2013). Semantic Text Classification for Supporting Automated Compliance Checking in Construction. *Journal of Computing in Civil Engineering*.

Singapore. (2013). Singapore Fire Code 2013 *http://www.scdf.gov.sg/content/scdf_internet/en/building-professionals/publications_and_circulars/fire-code-2013.html*.

Solibri. Solibri Model Checker. *http://www.solibri.com/solibri-model-checker/functionality-highlights.html*. Retrieved 03/01/2014, 2014

Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. *Automation in Construction, 53*(0), 69-82. doi: http://dx.doi.org/10.1016/j.autcon.2015.03.003

Solihin, W., Shaikh, N., Rong, X., & Poh, L. K. (2004). *Beyond Interoperability of Building Models: A Case for Code Compliance Checking*. Paper presented at the BP-CAD Workshop, Carnegie Melon University. https://www.researchgate.net/publication/280598933_BEYOND_INTEROPERATIBILITY_OF_BUILDING_MODEL_A_CASE_FOR_CODE_COMPLIANCE_CHECKING

Sowa, J. F. (1976). Conceptual graphs for a data base interface. *IBM Journal of Research and Development, 20*(4), 336-357.

Sowa, J. F. (1984). *Conceptual Structures - Information Processing in Mind and Machine*: Addison-Wesley Publishing Company.

Swaddiwudhipong, S., & Kog, Y. C. (2000). IT Strategy of the Singapore Construction Sector *Computing in Civil and Building Engineering (2000)* (pp. 635-642).

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

Wawan, S. (2016). *A Simplified BIM Data Representation Using a Relational Database Schema for an Efficient Rule Checking System and Its Associated Rule Checking Language.* (Ph.D. Dissertation), Georgia Institute of Technology.

Yurchyshyna, A., & Zarli, A. (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction, 18*(8), 1084-1098.

Zhang, C., & Beetz, J. (2015). *Model Checking on the Semantic Web: IFC Validation Using Modularized and Distributed Constraints*. Paper presented at the CIB W78 2015, Eindhoven University of Technology, October 27-29, 2015.

Zhang, J., & El-Gohary, N. (2012). Extraction of Construction Regulatory Requirements from Textual Documents Using Natural Language Processing Techniques. *Proc., Comput. Civ. Eng*, 453-460.

APPENDIX A - MORE EXAMPLES OF THE USE OF CG FOR VARIOUS RULES

10.1 A-1 Class-1 rules:

1. "*[F] 405.10 Standpipe system. The underground building shall be equipped throughout with a standpipe system in accordance with Section 905.*" (ICC, 2009)

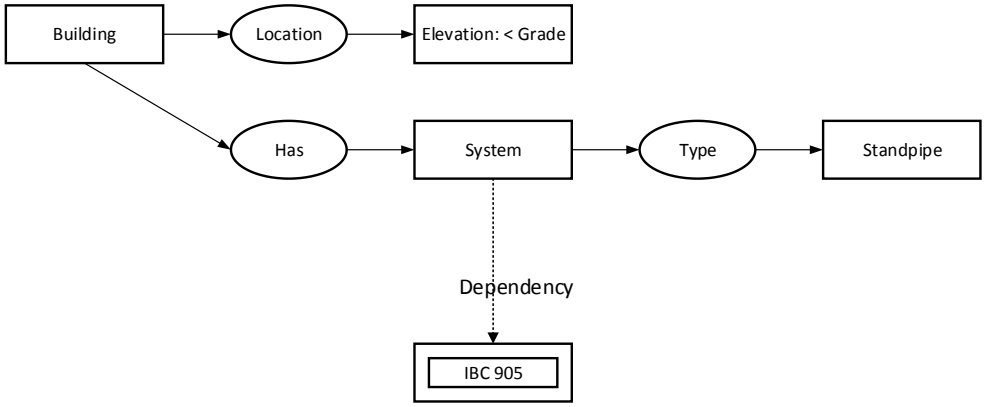


Figure 23 - CG for IBC/IFC 405.10 (ICC 2009)

2. *"Egress doors shall be of the pivoted or side-hinged swinging type"* (IBC 1008.1.2 – without exceptions) (ICC, 2009)

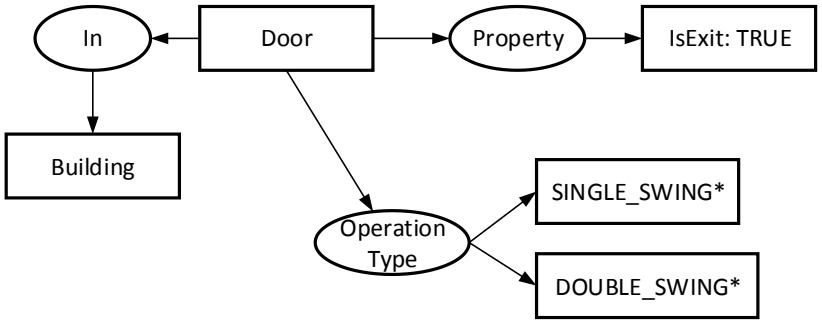


Figure 24 - CG for IBC 1008.1.2 (ICC 2009)

3. (ICC, 2009) "*504.2 Automatic sprinkler system increase. Where a building is equipped throughout with an approved automatic sprinkler system in accordance with Section 903.3.1.1, the value specified in Table 503 for maximum building height is increased by 20 feet (6096 mm) and the maximum number of stories is increased by one. These increases are permitted in addition to the building area increase in accordance with Sections 506.2 and 506.3. For Group R buildings equipped throughout with an approved automatic sprinkler system in accordance with Section 903.3.1.2, the value specified in Table 503 for maximum building height is increased by 20 feet (6096 mm) and the maximum number of stories is increased by one, but shall not exceed 60 feet (18 288 mm)or four stories, respectively.*

Exceptions:

1. *Buildings, or portions of buildings, classified as a Group I-2 occupancy of Type IIB, III, IV or V construction.*
2. *Buildings, or portions of buildings, classified as Group H-1, H-2, H-3 or H-5 occupancy.*

3. *Fire-resistance rating substitution in accordance with Table 601, Note d.*"

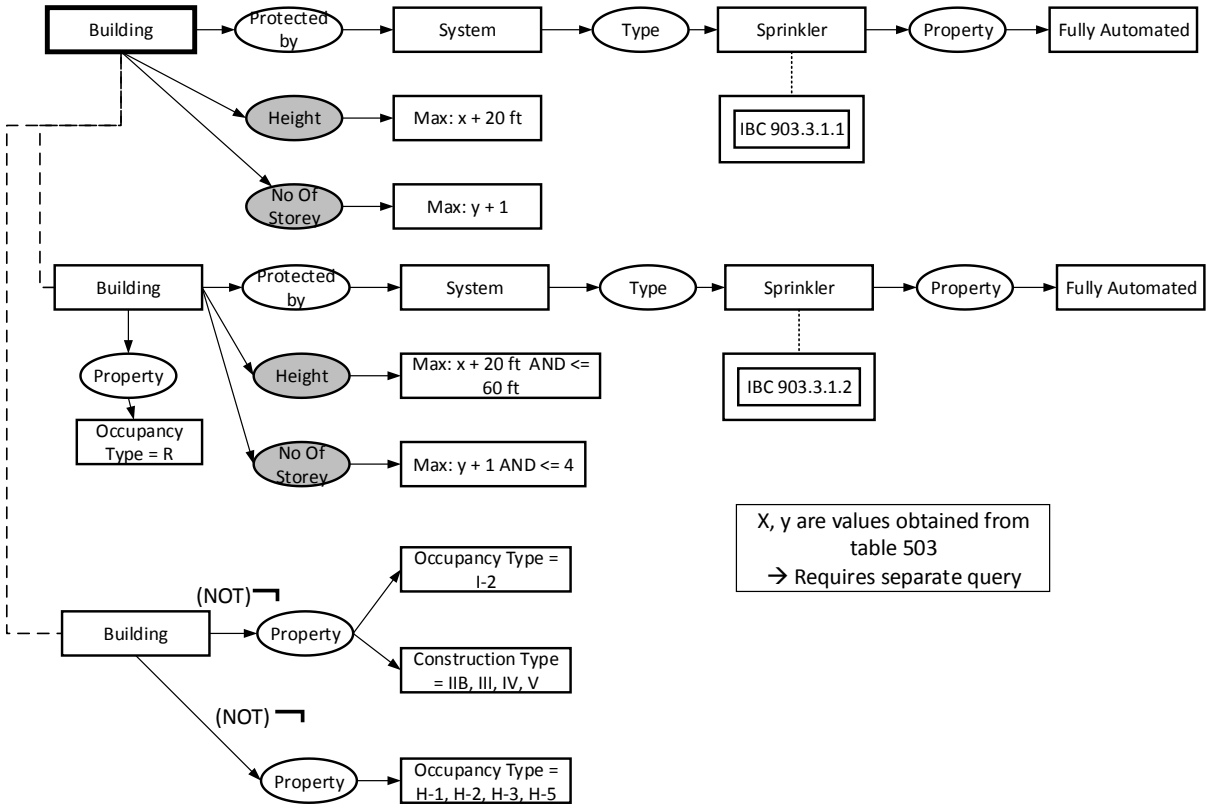


Figure 25 - CG for IBC 504.2 (ICC 2009)

10.2 A-2 Class-2 rules

1. "*1106.1 Required. Where parking is provided, accessible parking spaces shall be provided in compliance with Table 1106.1, except as required by Sections 1106.2 through 1106.4. Where more than one parking facility is provided on a site, the number of parking spaces required to be accessible shall be calculated separately for each parking facility.*

   Exception: This section does not apply to parking spaces used exclusively for buses, trucks, other delivery vehicles, law enforcement vehicles or vehicular impound and motor pools where lots accessed by the public are provided with an accessible passenger loading zone."

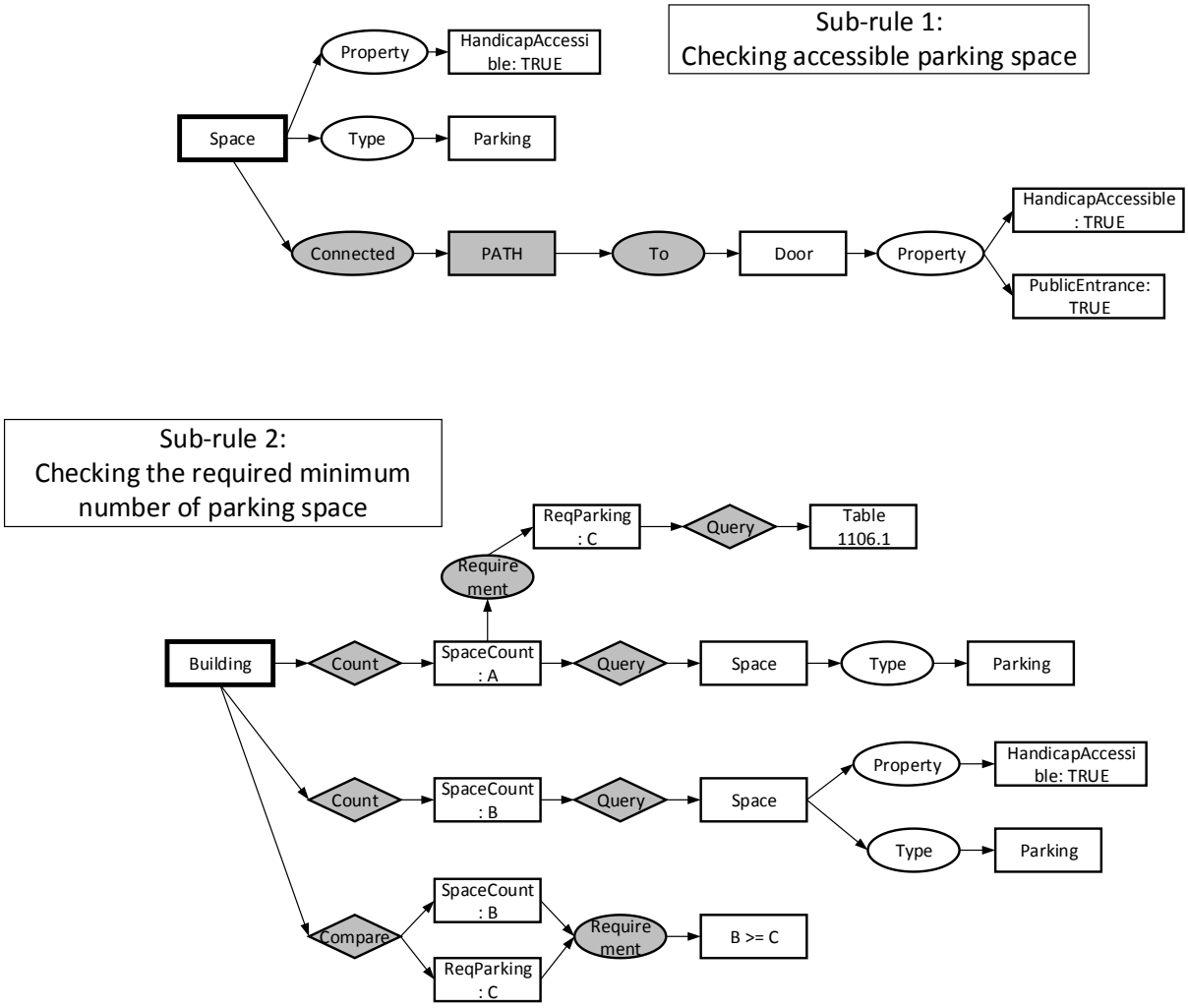


Figure 26 - CG for IBC 1106.1 (ICC 2009)

10.3 A-3 Class-3 rules

1. *"The judge's chambers are accessed from restricted circulation with convenient access to the courtrooms" (GSA)*

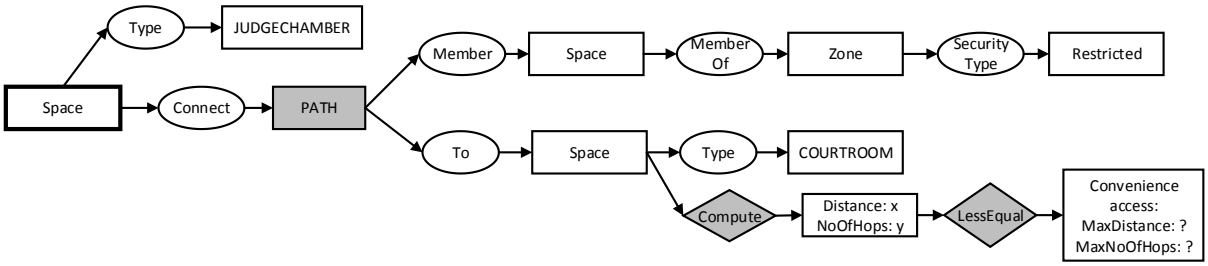


Figure 27 - CG for GSA Courthouse Rules on Accessibility of the Judge's Chambers

2. The complete IBC 1008.1.2 with all the exceptions (ICC, 2009):

1008.1.2 Door swing.
Egress doors shall be of the pivoted or side-hinged swinging type.

Exceptions:

1. *Private garages, office areas, factory and storage areas with an occupant load of 10 or less.*
2. *Group I-3 occupancies used as a place of detention.*
3. *Critical or intensive care patient rooms within suites of health care facilities.*
4. *Doors within or serving a single dwelling unit in Groups R-2 and R-3.*
5. *In other than Group H occupancies, revolving doors complying with Section* 1008.1.4.1*.*
6. *In other than Group H occupancies, horizontal sliding doors complying with Section* 1008.1.4.3 *are permitted in a means of egress.*
7. *Power-operated doors in accordance with Section* 1008.1.4.2*.*
8. *Doors serving a bathroom within an individual sleeping unit in Group R-1.*
9. *In other than Group H occupancies, manually operated horizontal sliding doors are permitted in a means of egress from spaces with an occupant load of 10 or less.*

Doors shall swing in the direction of egress travel where serving an occupant load of 50 or more persons or a Group H occupancy.

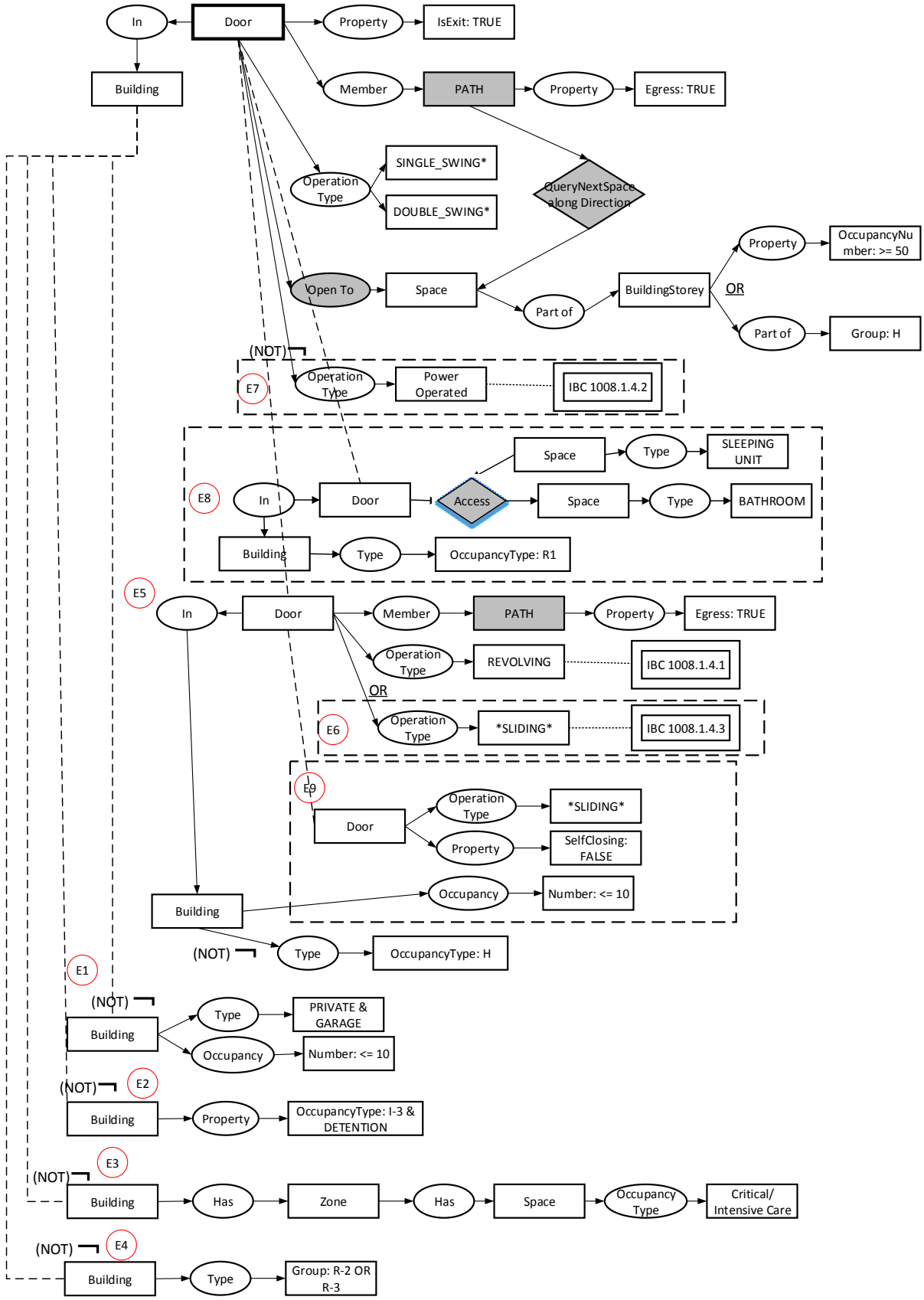


Figure 28 - CG for IBC 1008.1.2 Rule (ICC 2009)

3. 1027.1 General. (EXIT DISCHARGE)

Exits shall discharge directly to the exterior of the building. The exit discharge shall be at grade or shall provide direct access to grade. The exit discharge shall not re-enter a building. The combined use of Exceptions 1 and 2 below shall not exceed 50 percent of the number and capacity of the required exits.

Exceptions:

1. *A maximum of 50 percent of the number and capacity of the exit enclosures is permitted to egress through areas on the level of discharge provided all of the following are met:*
   - *1.1. Such exit enclosures egress to a free and unobstructed path of travel to an exterior exit door and such exit is readily visible and identifiable from the point of termination of the exit enclosure.*
   - *1.2. The entire area of the level of exit discharge is separated from areas below by construction conforming to the fire-resistance rating for the exit enclosure.*
   - *1.3. The egress path from the exit enclosure on the level of exit discharge is protected throughout by an approved automatic sprinkler system. All portions of the level of exit discharge with access to the egress path shall either be protected throughout with an automatic sprinkler system installed in accordance with Section 903.3.1.1 or 903.3.1.2, or separated from the egress path in accordance with the requirements for the enclosure of exits.*
2. *A maximum of 50 percent of the number and capacity of the exit enclosures is permitted to egress through a vestibule provided all of the following are met:*
   - *2.1. The entire area of the vestibule is separated from areas below by construction conforming to the fire-resistance rating for the exit enclosure.*
   - *2.2. The depth from the exterior of the building is not greater than 10 feet (3048 mm) and the length is not greater than 30 feet (9144 mm).*
   - *2.3. The area is separated from the remainder of the level of exit discharge by construction providing protection at least the equivalent of approved wired glass in steel frames.*
   - 2.4. *The area is used only for means of egress and exits directly to the outside*.
3. *(Omitted)*
4. *(Omitted). Exceptions 3 and 4 are omitted in this example because they are practically separate rules.*

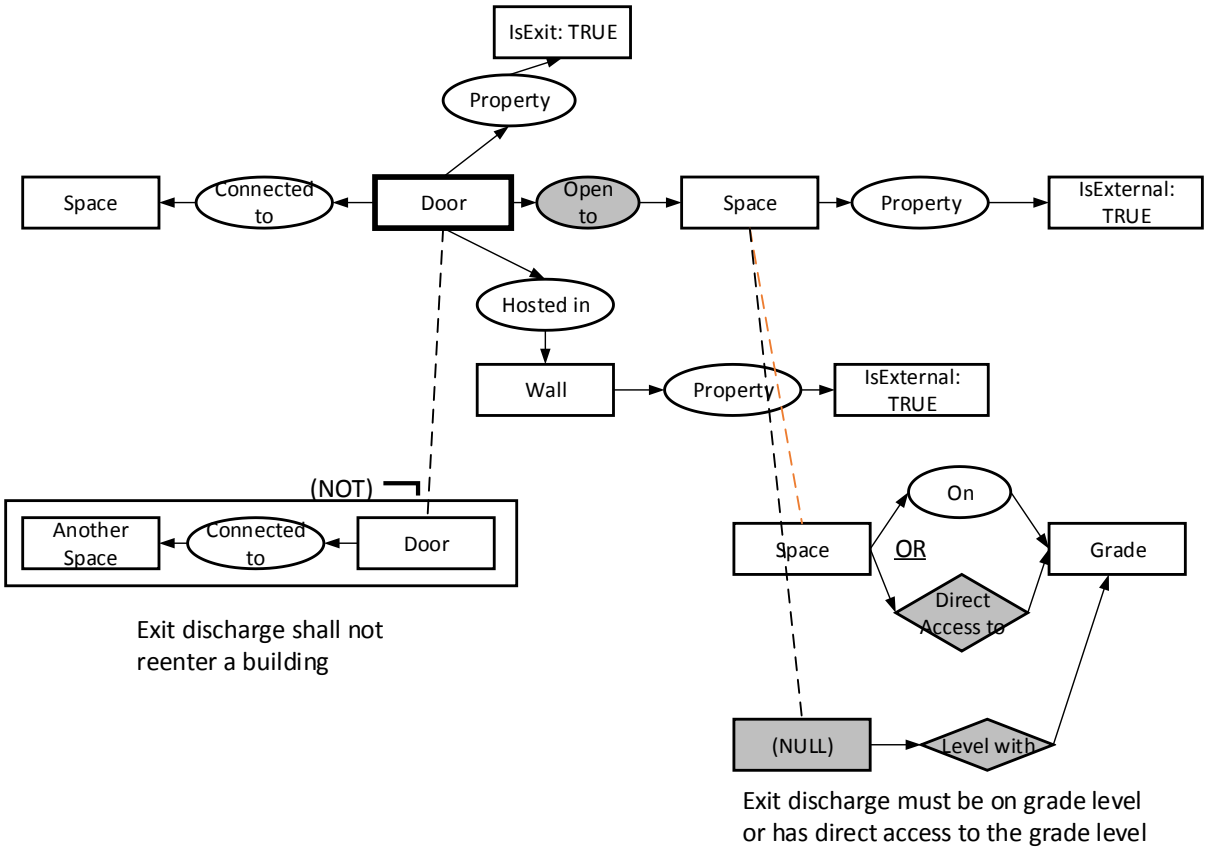


Figure 29 - CG for the IBC 1027.1 Main Rule (ICC 2009)

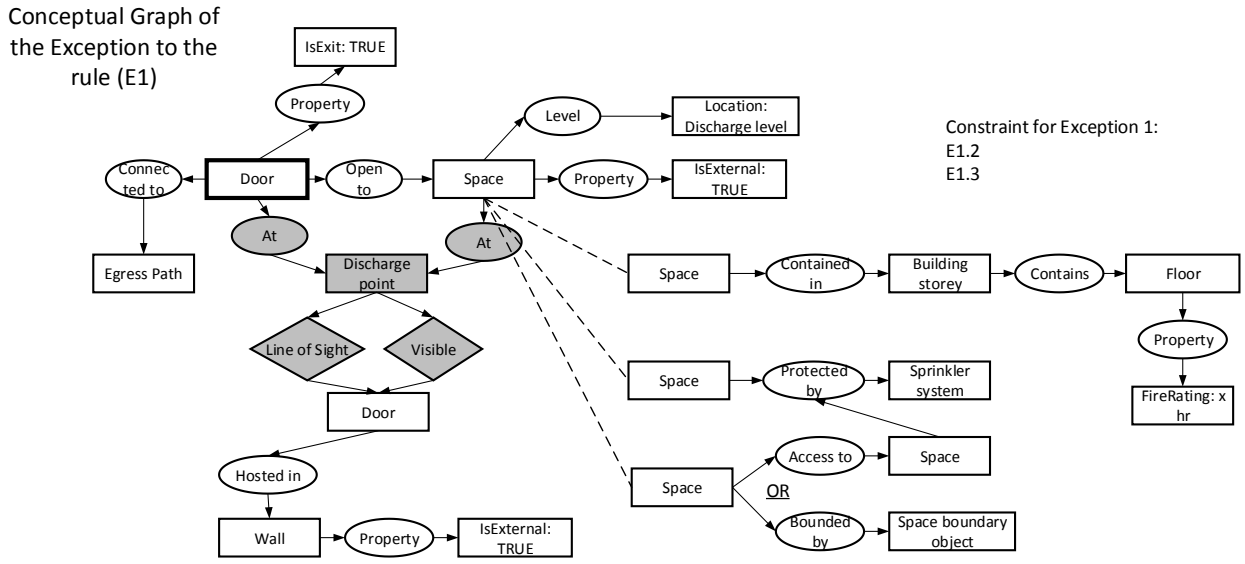


Figure 30 - CG for IBC 1027.1 Exception Rule no. 1 (ICC 2009)

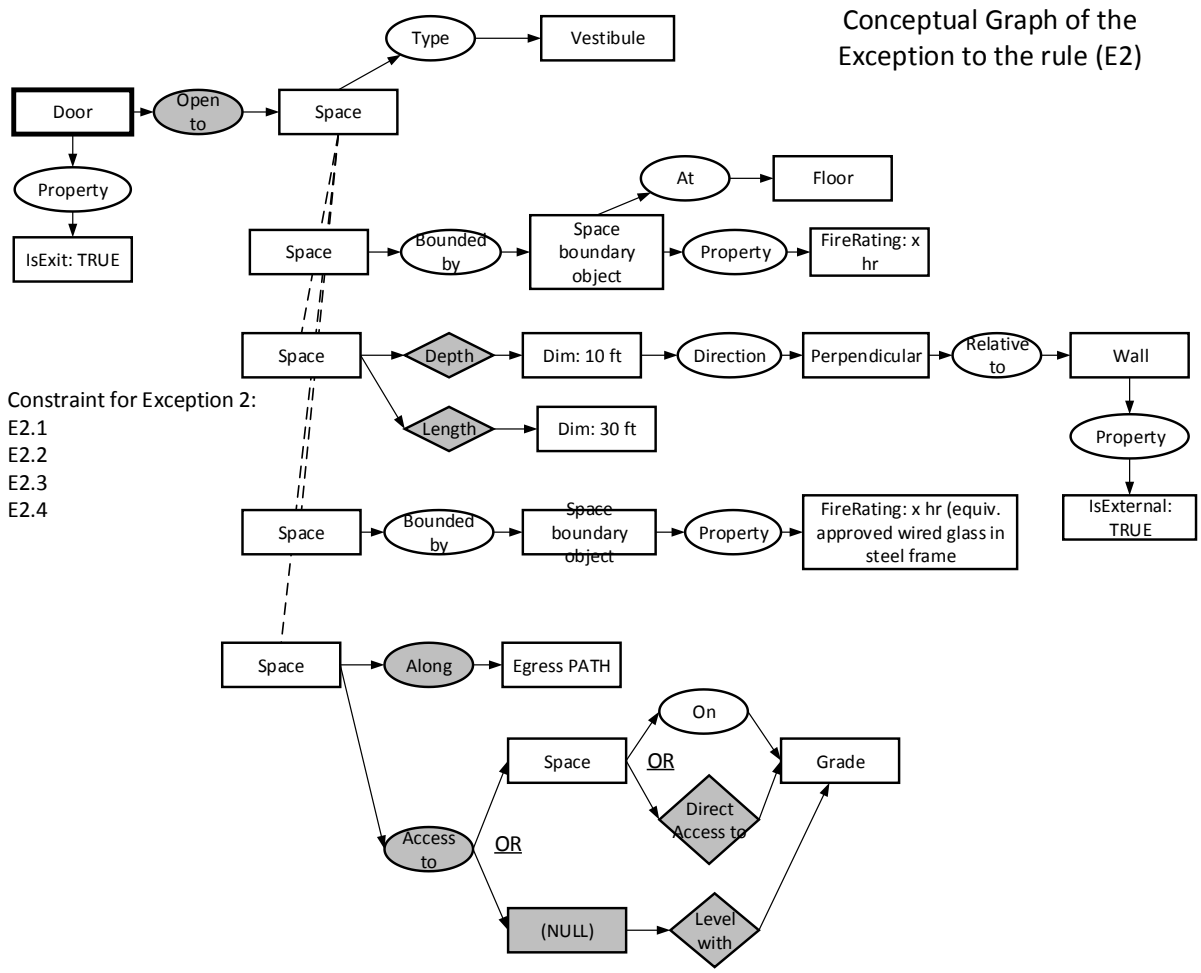


Figure 31 - CG for IBC 1027.1 Exception Rule no. 2 (ICC 2009)