E-Journal of Advanced Maintenance Vol.8-2 (2016) 13- 22
Japan Society of Maintenology

Building Ontology for Fire Emergency Planning and Support

Noel Nuo Wi TAY[1] , Naoyuki KUBOTA[1] and János BOTZHEIM[1]

[1] *Tokyo Metropolitan University, 6 Chome-5-4 Asahigaoka, Hino, Tokyo 191-0065, Japan*

ABSTRACT

Risk management and maintenance operation needs planning on emergency occurrences like a fire, which require information on building state. During fire emergencies, parts of a building may have higher risks compared to other parts of the building. This distribution of risks depends on the location of the fire, placement of hazardous materials in the building, condition of doors and windows around the building, vents and so on. Before getting to safety, one needs to know where safe places are, and how to approach them safely. This can be difficult in a large building complex. As the aforementioned attributes can be seen as objects and classes, they can be assigned their semantic meanings. This includes the description of the rooms, and their relationships with every other room, such as their adjacency. The building can be modeled by OWL description logic, which define the building as a graph, where objects are classified into classes, with properties to connect between them. The generated graph via description logic can be used for further inference for more data and for querying. With this approach, this work aims to build the building ontology to provide support during fire emergency. Case studies are done to infer smoke propagation and determination of escape outlet, which can provide the basis for additional query and inference.

KEYWORDS

Ontology, Description logic, fire emergency, knowledge representation

Article history:
Received 8 April 2016
Accepted 11 July 2016

ARTICLE INFORMATION

1. Introduction

Planning and simulating occurrences of emergencies given the actual state of a building like warehouse or factory plant is important for risk management and maintenance operation. Due to such activities are generally non-routine and incur downtime, planning has to be done carefully but quickly. By equipping the building with ambient intelligence and knowledge representation, well-informed information can be obtained easily by the workers. Such infrastructures are also useful in actual emergencies for both escapees and rescuers.

Recent progress in information processing and communication technology has made possible the creation of smart home, smart environments and office buildings that is able to increase comfort, security and efficiency. Information system has been widely used in the field of maintenance, designing and manufacturing for factories [2]. Ambient intelligence played a big part, which has the aim of surrounding building inhabitants with intelligence and unobtrusive sensors and actuators to acquire knowledge, adapt and dispense favorable services [1]. Mobile devices are utilized and objects (including non-electronic objects) are assigned with RFID tags that carry their information essential for the construction of knowledge structure and inference.

This is further enhanced by the progress of web technologies and the introduction of the semantic web, where various tools for knowledge representation, semantic annotations and querying are developed and standardized by the World Wide Web Consortium (W3C). This progress also brings forward the Internet of Things (IoT), where everyday objects are equipped with embedded data and communications capabilities. With this, IoT realizes the concept of pervasive and ubiquitous computing that is crucial for ambient intelligence.

Semantic web is introduced by Tim Berners-Lee, which is an extension of the current web where web information are endowed with semantics such that they can be operated and processed by machines [3, 4]. Substantial amount on research has been done that utilized Semantic Web

Technologies to build intelligent applications in various domains [7, 8, 9]. They have been applied to domotic applications [10, 11], agent communication [12] and emergency management [13] that exploits the knowledge structure built up via tools such as the ontology markup language (DAML, OWL), rule mark up language (SWRL), querying tool SPARQL and RDF schema, all of which are endorsed by W3C.

This work aims to extend the use of description logic and querying tools introduced by the field of semantic web to the construction of knowledge representation for emergency and safety purposes. In a large building complex with large number of rooms and object distribution, situation can be chaotic given a disaster situation such an earthquake, which can lead to fire in multiple locations of the building. Given the layout of the building and arrangement of crucial objects like those that are combustible, and also depending on the state of doors and windows, locating danger zone and escape outlet can be hard. The spread of smoke and gases, which closely depends on whether certain doors are opened/closed, further complicate the situation. Information on the possible danger zone in times of emergency is thus crucial. Given a building that is equipped with sensors to realize ambient intelligence and knowledge structure of the building, such information can be obtained instantaneously through description logic reasoning. Objects in the building can be assigned semantic meanings. This includes the description of the rooms, and their relationships with every other room, such as their adjacency. Description logic is used to define the building as a graph, where objects are classified into classes, with properties to connect them. The graph generated via description logic can be used for further inference for more data, and also be used for querying.

Description logics (DLs) are a family of knowledge representation languages used for ontological modeling [5]. DLs are equipped with formal semantics that allows the exchange of ontologies without ambiguity as to their meaning. They also enable inference to be made given the axioms stated in the ontology. There are different kinds of DLs. We choose OWL Web Ontology Language to model the building ontology due to its compatibility with the semantic web endorsed by W3C (thus, abundant tools are available), and have the benefit of decidable inference that prevents on-going looping. The advantage of using OWL DL is that they are always guaranteed to terminate during inference, unlike using pure First Order Logic (FOL).

By modeling a building as a semantic graph described via OWL DL, syntactic risks of every part of the building (given every part are treated as an individual object) can be inferred using an inference engine or reasoner. With the reasoner, inference can be made in real-time with current situation. This is important, as the world is dynamic, such as doors opening and closing or locked, and hazardous materials being moved around.

As situations and the effects of fire are highly unpredictable, axioms obtained from description logic are of course not sufficient to model the real environment. Building fire emergencies need to take into account not just the surrounding, but also the flow-of-influence for object states and distributions, which requires logical constructs. One can resort to densely modeled rules (which is extremely tedious and difficult to expand) or inferred from probabilistic or analogous models like probabilistic graphical model and SVM respectively (though they cannot directly support logical constructs and notion of causality). Our intention is to provide the latter with representations that are generated from the logical constructs of the description logic. Case studies done are to demonstrate the influence of flow due to object relationships inferred via OWL DL, where it is decidable. Learning models can use the stable representation to acquire more accurate depiction of the environment.

2. Building Knowledge Base

2.1. Building Ontology

Ontology is setup such that knowledge can be represented according to its schema, which can support reasoning on possible syntactical dangers and escape outlets. In this work, emphasis is given to dangers from fire emergency. In such a case, building layout and location of combustible objects as well as possible escape outlets are important in determining the magnitude of danger of different locations of the building.

Therefore, the building ontology is built according to the building layout (rooms, doors, windows), and the classification of objects (package, combustible materials). Ontology is built based on the OWL DL language. Due to the limitation imposed on OWL in order to ensure that inference can always terminate, certain crucial functions such as implication rules cannot be directly modeled using the language. In this case, rules are normally modeled through SWRL. In this work, we use the query language SPARQL (which is the query language endorsed by W3C) to perform the work of SWRL. SPARQL is a descriptive query language that is able to support the implementation of rules and assertion of new axioms in the knowledge base [6]. But care should be exercised when using SWRL or SPARQL to avoid inconsistency of the ontology.

To simplify notation in this paper, property, which is part of the Resource Description Framework (RDF) triple, will sometimes be represented in predicate form. For example, an RDF triple "*Adam isA Boy*", where *isA is a property,* can be represented as predicate *isA(Adam, Boy)*. This predicate will return true if the triple "*Adam isA Boy.*" exists in the knowledge base, and false otherwise. Besides that, a class can be shown as concept assertions, where triple "*Dog rdf:type Animal.*" can be represented as *Animal(Dog)*.

Also, for new triple assertion, where SPARQL is used, in this paper, First Order Predicate is represented for better readability. For example, given the following formula:

isA(?a,Boy) → isA(?a,Human)

which means that if an instance is related to *Boy* via *isA*, then it is also related to *Human* via *isA*. This formula represents SPARQL Construct query shown below:

```
CONSTRUCT {?a :isA :Human.}
WHERE {?a :isA :Boy}
```

More information on the modeling of ontology can be obtained from [6].

2.2. Object Classification

For object classification, objects are assigned to classes that are relevant to their properties. It is assumed that the objects involved contain RFID tags with their unique ID that can be easily read (although other types of identification method that can also be used for tracking are equally acceptable). The intention is to provide all objects with a unique ID, and at the same time, be able to be tracked such that their location is known every time. Given their IDs, their properties can be obtained from the building knowledge base, and thus, inference ensued. Given objects RFID are detected, they are then represented as instance, and are assigned to a class via rdf:type property. Given this, objects can be easily assigned to their attributes (combustible, explosive, acidic etc.), functions (product2, fire extinguisher), locations (room1, kitchen) and package group. For simplicity, only relationships that are relevant to the case studies are described.

Every object is represented as a node in the semantic network graph, which is also an instance. Objects will be assigned IDs via property *hasID*.

Certain objects may come in groups with other objects, such as materials in a luggage or a container. Since the group is also an object, it has its own reified node. Objects that belong to the group are related to it through the property *inGroup*. Group objects have their own unique class to encase all the objects that belongs in it. Therefore, for every group object, a class is created that is equipped with the restriction (in Turtle):

```
[a owl:Restriction;
owl:onProperty :inGroup;
owl:hasValue G].
```

where *G* is the instance of the group. This means, any instance that is related to the instance *G* through property *inGroup* will be assigned under the unique class corresponding to *G*.

Objects will also be assigned attributes. Attributes that are relevant to the subsequent case studies are being combustible, explosive and acidic. These attributes are represented as classes, where objects with such attributes will be assigned to the relevant class. Note that these classes may overlap. The three attributes (*combustible, explosive, acidic*) are subsumed by the class *dangerousMaterials*.

To represent the location of objects, every object instance is related to a location instance through property *locatedAt*. Care needs to be taken when specifying locations for objects in groups, such that all of them, including the group itself, are pointing to the same location node.

It is assumed that there are means to detect humans in the rooms, either from direct human detection via camera or RFID. A room is assigned under *haveHuman* class if humans are detected.

Knowing whether there are humans and their locations in the building is important for rescuers to take further actions, as well as determining whether the humans are under trapped position or not. For the latter, it can be represented by the following rule:

haveHuman(?r) ∧ ¬validEscape0(?r) →rescue(?r)

The rule states that if a room has humans and that it is not an escape path, then it requires rescue by assigning the room under *rescue* class. Details on *validEscape0* are given in Section 2.4.

2.3. Building Layout Modeling

For building layout, ontology needs to be built such that room adjacency and their relationships with any other rooms can be obtained. These relationships should also be influenced by the opening/closing of doors, location of escape outlet and where dangers are located.

A door must be associated with two rooms. Its relationship with a room is via property *isDoor*. A door belongs to class *doorOpen* if it is open. Therefore, before inference begins, axioms should be asserted for every door that is open. Doors that are not assigned to this class are considered closed.

Two rooms are adjacent if at least one of the doors between them is open. Adjacency between two rooms is defined by the property *isAdjacent*, which has a symmetric property (if room 1 is adjacent to room 2, so is the case for the other way round). Adjacency axiom can be asserted by:

isDoor(?d,?r1) ∧ isDoor(?d,?r2) ∧ doorOpen(?d) → isAdjacent(?r1,?r2)

Another alternative to the above mentioned rule is to use restriction from cardinality on the property *isAdjacent*. This eliminates the need to explicitly assert axioms of adjacency as it can be taken care of by the inference engine, but preliminary test shows that this approach is very slow.

Property *isConnected* is defined by the rule:

isAdjacent(?r1,?r2) → isConnected(?r1,?r2)

Property *isConnected* is not equivalent to *isAdjacent*. *isConnected* is endowed with transitive property. Whereas *isAdjacent* only links two rooms that are next to each other given that there is a path between them, *isConnected* relates a room (denoted as RoomA) to any rooms that are related to any other rooms that are related to RoomA via *isConnected*.

A room's relationship with windows is made through the property *isWindow*. In this work, no windows between rooms are assumed, except for windows the links rooms with the outside.

There is a node that reifies the state of being outside of the building, which is the *outside* node. It is treated the same as any other rooms, except that it provides the point of an escape outlet. Windows and doors that leads to outside of the building, given that they are open, will cause the corresponding room to be adjacent to the outside, which is described as:

isDoor(?d,outside) ∧ isDoor(?d,?r) ∧ doorOpen(?d) → isAdjacent(?r,outside)

isWindow(?w,r) ∧ windowOpen(?w) → isAdjacent(?r,outside)

Given the above rules, since *outside* provides the point of an escape outlet, the following rule describes this property:

isAdjacent(?r,outside) →escape5(?r)

escape5 indicates that the room assigned under it is a potential escape outlet, but not under full certainty, as danger like fire may occur in the same room. The details of the class *escape5* will be given in Section 2.4.

2.4. Influence from Adjacency

Given two rooms are adjacent, certain influence of danger or determining whether the current position will lead to an escape outlet can be inferred from the information of the adjacent room. In this sub-section, two influences from adjacency will be explained, which are 1) the spread of smoke 2) determination of escape outlet

In a building, it is assumed that there are sensors in place that detects fire and smoke. Any room with danger detected will be assigned under the class *roomDanger*. Danger can be known from the sensors on fire and level of smoke, and also whether there are hazardous materials around. The remaining rooms will be assigned *nonDanger* by the following rule:

room(?r) ∧ notAssigned(roomDanger,?r))→ nonDanger(?r) where *notAssigned(a,b)* means if *b* is not assigned under *a*. Predicate *room(a)* returns whether *a* is of class *room* or not. Every instance preceded by 'R' in Fig. 1. is of class *room*.

Smoke level consists of multiple levels, depending on the severity. The number of levels depends on the individual choosing. In this paper, 6 levels are assigned, starting from level 0 to level 5, where level 5 has the highest severity. A room with a certain level of smoke is assigned to the corresponding class, denoted by *levelClass'n'*, where 'n' is the number of levels. Since there are 6 levels, the classes are *levelClass0*, *levelClass1, levelClass2, levelClass3, levelClass4* and *levelClass5*. It can be safely assumed that a class of lower level subsumes the higher class. For example, a room with smoke level of 5 will have at least a smoke level of 4, and so on.

Given this assumption, flow of influence due to adjacency can be modeled by imposing restriction on the level class as {*levelClass'n'* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass'n+1'*}, which in Turtle is:

levelClass'n' owl:equivalentClass
[a owl:Restriction;
owl:onProperty :isAdjacent;
owl:someValuesFrom levelClass'n+1'].

In this case, the smoke level of a room can cause rooms adjacent to it to be assigned to the class of lower smoke level.

As mentioned before, determining escape outlet can also be influenced by information of the adjacent room. Like wise to smoke level, we will assign 6 levels to signify how near a particular room is to the nearest escape outlet, where the level class is denoted as *escape'n'*, namely *escape0, escape1, escape2, escape3, escape4* and *escape5*. The higher the level, the nearer the room is to an escape outlet. Also, lower level class subsumes the higher level class. But the flow of influence is not as direct as that in the case of smoke. A room that is deemed dangerous should not propagate influence regarding escape outlet, which means, flow of influence only works for those rooms assigned under the class *nonDanger*. Lets denote the intersection between class *escape'n'* and *nonDanger* as *validEscape'n'*. Restrictions is then applied to this class instead as (in Turtle):

escape'n' owl:equivalentClass
[a owl:Restriction;
owl:onProperty :isAdjacent;
owl:someValuesFrom validEscape'n+1'].

2.5. Multistage Inference

Inference for the state of the building cannot be done in one stage, where certain axioms can only be asserted given previous axioms, such as the determination of escape outlet that needs information on the *nonDanger* class. In this work, the number of stages is limited to two.

In the first stage, the core ontology is asserted, which includes the symmetric property of *isAdjacent*, restrictions on *levelClass'n'*, subsumption for *levelClass'n'*, etc. The core assertions for the first stage are shown in Table 1. In this stage, axioms asserted from sensors and assertion of adjacency from opened doors are also performed, which are:

1) *isDoor(?d,?r1) ∧ isDoor(?d,?r2) ∧ doorOpen(?d) → isAdjacent(?r1,?r2)*

2) Assigning rooms to *roomDanger* if sensors pick up any danger signal

Table 1 Assertions in first stage inference

No.	Assertion
1	Restriction to *levelClass4* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass5*
2	Restriction to *levelClass3* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass4*
3	Restriction to *levelClass2* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass3*
4	Restriction to *levelClass1* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass2*
5	Restriction to *levelClass0* with *owl:someValuesFrom* on property *isAdjacent* to *levelClass1*
6	*validEscape5* as intersection of *escape5* and *nonDanger*
7	*validEscape4* as intersection of *escape4* and *nonDanger*
8	*validEscape3* as intersection of *escape3* and *nonDanger*

9	*validEscape2* as intersection of *escape2* and *nonDanger*
10	*validEscape1* as intersection of *escape1* and *nonDanger*
11	*validEscape0* as intersection of *escape0* and *nonDanger*
12	Restriction to *escape4* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape5*
13	Restriction to *escape3* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape4*
14	Restriction to *escape2* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape3*
15	Restriction to *escape1* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape2*
16	Restriction to *escape0* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape1*
17	Restriction to *escape0* with *owl:someValuesFrom* on property *isAdjacent* to *validEscape0*
18	$escape5 \subseteq escape4 \subseteq escape3 \subseteq escape2 \subseteq escape1 \subseteq escape0$
19	$levelClass5 \subseteq levelClass4 \subseteq levelClass3 \subseteq levelClass2 \subseteq levelClass1 \subseteq levelClass0$
20	$Combustible \cup explosive \cup acidic \subseteq dangerousMaterials$
21	*owl:SymmetricProperty* for *isAdjacent*
22	*owl:TransitiveProperty* for *isConnected*

For the second stage,
room(?r) ∧ notAssigned(roomDanger,?r))→ nonDanger(?r)
haveHuman(?r) ∧¬validEscape0(?r) →rescue(?r)
are run. Inference after this will show possible escape outlet for every room.

3. Case Studies

Case studies are performed to demonstrate the applicability of the designed building ontology to generate semantic graphs for emergency situation. The main focus is on the detection of danger zones and determining escape outlet given the point of disaster, distribution of objects, state of doors and possible escape outlets. For this case study, FaCT++ [14] is used as the description logic inference engine. Rules and axiom assertions are implemented through SPARQL, which, in this work, is uniquely written to work with FaCT++.

3.1. Building Layout

Fig.1. shows the building layout used for the case studies. The figure shows rooms (indicated by 'R'), doors (in blue indicated by 'd') and window (in red indicated by 'w').

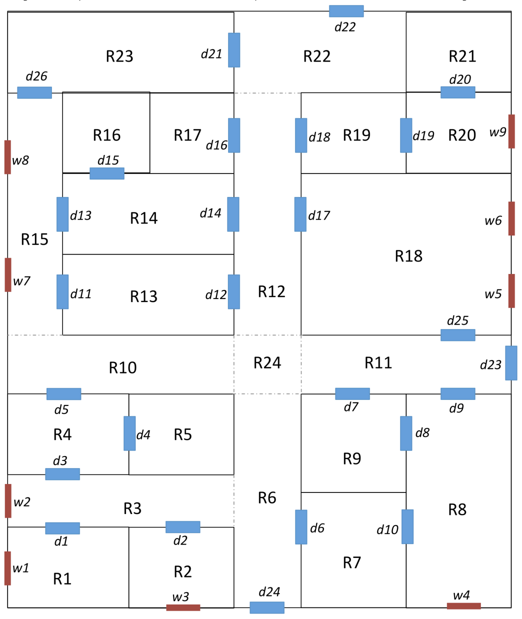

Fig. 1. Building Layout. Blue rectangle indicates doors, red rectangle indicates windows

Given the windows and doors allocation, room R1, R2, R3, R6, R8, R11, R15, R18, R20 and R22 have the potential of being escape outlets if the doors or windows are opened.
Three configurations are demonstrated.
Configuration 1:
Fire occurs in corridor R10. There is an object that is combustible in room R13. Doors d11, d12, d13, d14 and d23 are opened. All windows are closed.
Configuration 2:
Fire occurs in corridor R12 and R11. There is acidic material in room R4. Doors d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d21, d22, d23 and d26 are opened. Window w6 is open.
Configuration 3:
Fire occurs in R3 and R12. All doors are open, and all windows are closed. There are humans in R1, R21 and R22.

3.2. Smoke Hazard Visualization

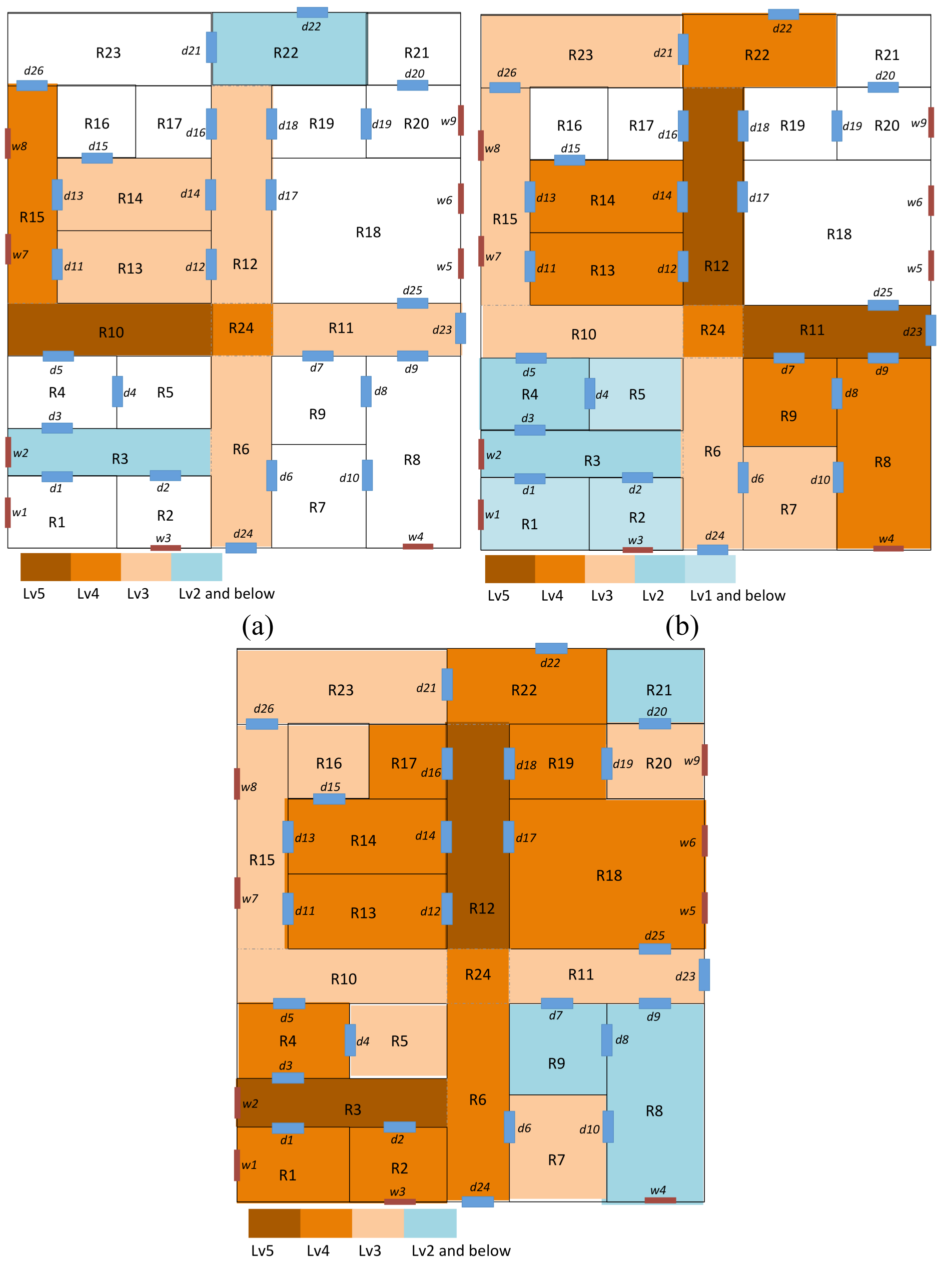
(a)
Lv5 Lv4 Lv3 Lv2 and below
(b)
Lv5 Lv4 Lv3 Lv2 Lv1 and below
(c)
Lv5 Lv4 Lv3 Lv2 and below
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24
d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18 d19 d20 d21 d22 d23 d24 d25 d26
w1 w2 w3 w4 w5 w6 w7 w8 w9

Fig. 2. Visualization of smoke spread for Configuration one (a), two (b) and three (c)

In times of fire, it is generally useful to know how the smoke will spread given the condition of doors. This can help escapees and rescuers evaluate the danger of the particular section of the building. Test is run on the three configurations stated in Section 3.1. Fig.2. shows the visualization of smoke spread obtained from inference. Smoke spread information can be obtained at the first stage of multistage inference.

For configuration 1, fire starts at R10. Smoke is at the highest level in R10, thus, R10 is assigned under class *levelClass5*. Smoke spread according to the state of opened doors. Given the adjacent rooms, smoke level is inferred for rooms surrounding the point of fire. Rooms that are unmarked by colors indicating different levels in the figure are considered safe from the smoke.

For configuration 2, fire starts in R12 and R11. Configuration 2 has more opened doors compared to configuration 1, thus, the spread of smoke is wider. In this configuration, the spread of smoke is important in determining which rooms require more assistance or the control of vents. This can be observed from room R23, R15 and R10, where there is a constant level of inferred smoke level. Whereas, for rooms R1, R2, R3, R4 and R5, the smoke level is significantly lower. In this case, this information is important for both the escapees and rescuers to evaluate the overall situation in terms of smoke propagation.

Since all doors in configuration 3 are opened, visualization shows that the smoke will spread throughout the building.

3.3. Escape Outlet

After the second stage of the multistage inference, escape outlet can be obtained. Fig.3. shows the visualization of escape path obtained from inference. Rooms that are not shaded are considered not part of the escape path.

For smoke spread information obtained in Section 3.2, smoke propagation does not give details of where a person should be headed to. Besides, distribution of hazardous materials will complicate the situation, causing people to take inefficient or wrong routes to safety. Information on such path, which takes into account points of danger, is important as it assists in deciding directions to take in order to reach a potential escape outlet.

Configuration one only has d23 opened, leading to the outside. Therefore, this configuration only has one escape outlet. Due to fire occurring in R10, and the presence of hazardous materials in R13, the figure clearly shows the desirable path for people in R15 is to go through R14 instead, which can be inferred via building ontology.

Configuration two shows two escape outlet of level 5, which is R22 (has a door to the outside) and R18 (has a window). Although d23 is opened, which makes R11 potentially a level-5 escape outlet, but due to fire occurring in R11, this information is not propagated further. For people in rooms other than R18, their only outlet is through d22 in R22. From the result, human may know that they need to approach R15 in order to reach the escape outlet safely, as other ways are blocked. Path via R4 is not being considered a path as there are hazardous materials inside, thus, people from R1, R2 and R3 should approach R6 to R24 and finally R10 to reach R15 to ensure their safety.

Result from configuration two clearly illustrates the point as stated earlier that smoke spread does not provide sufficient information on the escape route. From Fig.2., it seems R1, R2, R3, R4 and R5 are the places to be to avoid smoke. But these locations are dead end, and therefore, from Fig.3., it is wiser to follow a safe route to the nearest escape outlet.

Configuration three has humans in rooms R1, R21 and R22. From the result, R1 and R21 are assigned under *rescue* class because both these rooms are not escape paths. R1 is blocked by the fire in R3, and R21 is blocked by the fire in R12. Humans in both these rooms cannot easily reach any potential escape outlets. This information is crucial for rescuers to assess the situation and reach the victims. For humans in R22, since the room is an escape path, it is not assigned under *rescue* class.

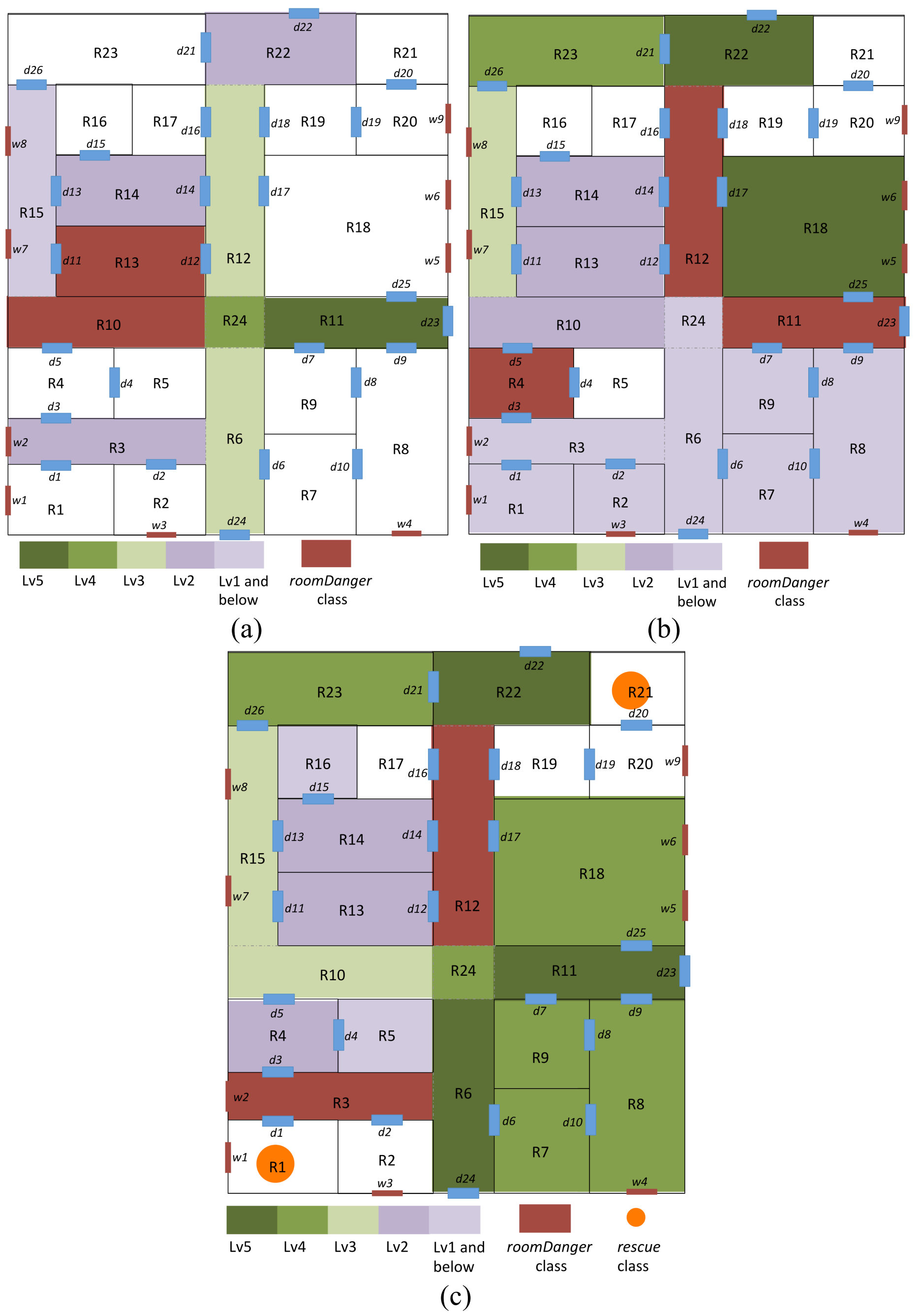

Fig. 3. Visualization of escape outlet for Configuration one (a), two (b) and three (c)

4. Conclusion

Two-stage inference of building ontology is built via description logic to support human in times of fire emergency for escapees and rescuers. Given the state of doors and windows and also the distribution of hazardous materials, as well as the general layout of the building, human judgments are negatively affected that is aggravated by the chaos of the situation. Building ontology can provide information on smoke propagation and determination of escape outlet syntactically, which is obtained through multistage inference that is decidable. Case studies on a complex building layout to provide such crucial information demonstrate the effective modeling of the influence flow in times of emergency. Statistical learning models can use these stable logical constructs for refined inference.

As the current work is only limited to two stage inference for demonstration, in actuality, it can be extended to higher number of stages to augment much more complex axioms that can aid in situational classification. Besides, buildings can be partitioned into finer sections for more accurate modeling, but this comes at a cost of processing time. Methods should be devised to divide large building complexes to process them separately, instead of processing the ontology of the whole building.

Current work concentrates on fire emergency. But since this application depends on the ontology that is specific, given further modifications, this approach can be applied to wide range of applications by incorporating information of relationships between objects and factors. Future plans are to be able to apply this approach to risk management and maintenance for actual plants, complemented with statistical learning models.

References

[1] N. Shadbolt: "Ambient Intelligence", IEEE Intelligent Systems, Vol.18, pp. 2-3 (2003).

[2] A. Nakajima, T. Onishi, T. Sawaragi: "Building Interoperable System Enabling Failure Information Knowledge Sharing Between Designing/Manufacturing and Maintenance Departments Focusing on Information Architecture Differences", E-Journal of Advanced Maintenance, Vol.2, pp. 120-134 (2010).

[3] T. Berners-Lee, J. Hendler, O. Lassila: "The Semantic Web", Scientific Amer., pp. 29-37 (2001).

[4] N. Shadbolt, T. Berners-Lee, W. Hall: "The Semantic Web Revisited", IEEE Intelligent Systems, Vol.21, No. 3, pp. 96-101 (2006).

[5] M. Krötzsch, F. Simancik, I. Horrocks: "A Description Logic Primer", arXiv:1201.4089 (2013).

[6] D. Allemang, J. Hendler: "Semantic Web of the Working Ontologist", Morgan Kaufman, ISBN 9780123859655 (2012).

[7] I.C. Hsu: "Extensible Access Control Markup Language Integrated with Semantic Web Technologies", Information Sciences, Vol. 238, pp. 33-51 (2013).

[8] C. De Maio, G. Fenza, M. Gallo, V. Loia, S. Senatore: "Formal and Relational Concept Analysis for Fuzzy-based Automatic Semantic Annotation", Applied Intelligence, Vol. 40, No. 1, pp. 154-177 (2013).

[9] A. Razzaq, K. Latif, H.F. Ahmad, A. Hur, Z. Anwar, P.C. Bloodsworth: "Semantic Security Against Web Application Attacks", Information Sciences, Vol. 254, pp. 19-38 (2014).

[10] D. Bonino, E. Castellina, F. Corno: "The DOG Gateway: Enabling Ontology-based Intelligent Domotic Environments", IEEE Trans. On Consumer Electronics, Vol. 54, No. 4, pp. 1656-1664 (2008).

[11] M. Ruta, F. Scioscia, E. Di Sciascio, G. Loseto: "Semantic-based Enhancement of ISO/IEC 14543-3 EIB/KNX Standard for Building Automation", IEEE Trans. Of Industrial Informatics, Vol. 7, No. 4, pp. 731-739 (2011).

[12] F.P. Pai, I.C. Hsu, Y.C. Chung: "Semantic Web Technology for Agent Interoperability: A Proposed Infrastructure", Applied Intelligence, Vol. 44, No. 1, pp. 1-16 (2015).

[13] Z.U. Shamszaman, S.S. Ara, I. Chong, Y.K. Jeong: "Web-of-Objects (WoO)-based Context Aware Emergency Fire Management Systems for the Internet of Things", Sensors, Vol. 14, No. 2, pp. 2944-2966 (2014).

[14] D. Tsarkov, I. Horrocks, "FaCT++ Description Logic Reasoner: System Description", Automated Reasoning, Vol. 4130, pp. 292-297 (2006).