Ontology-Based Understanding of Architectural Drawings

Lluís-Pere de las Heras, Oriol Ramos Terrades(✉), and Josep Lladós

Computer Vision Center, Universitat Autònoma de Barcelona,
Campus UAB, 08193 Bellatera, Barcelona, Catalonia, Spain
{lpheras,oramos,josep}@cvc.uab.es
http://www.cvc.uab.cat

Abstract. In this paper we present a knowledge base of architectural documents aiming at improving existing methods of floor plan classification and understanding. It consists of an ontological definition of the domain and the inclusion of real instances coming from both, automatically interpreted and manually labeled documents. The knowledge base has proven to be an effective tool to structure our knowledge and to easily maintain and upgrade it. Moreover, it is an appropriate means to automatically check the consistency of relational data and a convenient complement of hard-coded knowledge interpretation systems.

Keywords: Graphics recognition · Floor plan analysis · Domain ontology

1 Introduction

Graphical documents convey complex semantic concepts understandable by humans. This information is structured agreeing to a visual language; consisting of a vocabulary –graphical symbols– and a syntax –contextual relations–. Therefore, the semantic content expressed in a document is defined by the contextualized meaning of its structurally related symbols. Let us exemplify this fact by making an analogy to natural languages. The following two sentences are correct in terms of vocabulary and syntax:

People drive cars
Cars drive people

Even though both sentences consist of the same set of words, their structure –syntactical positioning– leads them to express completely different meanings. Moreover, given our natural knowledge of the language domain –the real world–, we can assert that one of the sentences expresses an unlikely event.

Alike to natural language comprehension, graphical document understanding requires the knowledge of the document domain. This knowledge defines the meaning of the compounding items in a determined context. For instance, the color, the shape, and the relative location of objects in a document agree to a defined visual knowledge in order to express desirable semantic concepts.

Therefore, to make computers able to understand graphical documents, we need to provide them with the appropriate tools to define, store, and employ this knowledge.

Ontologies are machine-interpretable specifications of conceptualizations [9]. They make explicit the description of concepts (classes), their attributes (properties), and mutual relationships that can exist in a domain. The domain definitions are written in formal languages with an expressive power close to the first-order logic; the language definition is independent to the data structure. Therefore, ontologies allow to describe a domain knowledge in a manner that it can be reused, incremented, and shared by disparate agents. Additionally, an ontological definition together with individual instances of the classes conforms a knowledge base that can be analyzed, queried, and classified semantically. Ontological definitions have already demonstrated their suitability in multiple Computer Vision scenarios, e.g. object categorization [14] and recognition [19], medical imaging [15], and natural image description [16]. Additionally, strongly related to our framework, Bhatt et al. present in [5] an ontological formalization of the architectural design domain that tries to link the earlier structural perception of an architect with the actual functionality of a design. In consequence, and given their properties, ontologies are convenient tools to express the domain of graphical documents.

In this paper we present a tentative exploration of the ontological modeling for graphical document understanding. More specifically, we have created a domain ontology of architectural drawings that allows us to perform semantic classification, retrieval, and validation of these documents. In Sect. 2, we introduce the floor plan knowledge base. Section 3 is devoted to overview the experiments performed. Finally, in Sect. 4 we conclude this paper.

2 Floor Plan Knowledge Base

We have created a knowledge base consisting of a formal definition of floor plan documents and a set real instances coming from both, automatic interpretation and manual annotation. This knowledge base has been created with the aim of filling the following intentions:

- To define specifically the semantics of our domain. We have created a floor plan ontology that permits us to describe formally the taxonomy of the concepts conveyed in floor plans, their properties, and relations.
- To permit the reutilization and maintenance of the domain. Since this is a long term project, the formal definition of the domain eases its maintenance; there is an independence between the interacting implementations and the ontology. Moreover, it allows to other agents, either human or automatic, to reuse and upgrade our definition at their convenience.
- To allow semantic reasoning with real data. The inclusion of instances agreeing the ontological framework allow to classify and validate them regarding the definition of the concepts, attributes, and relations.

The knowledge base is defined using the Web Ontology Language OWL2 [11] on the Protégé5 [3] ontology editor. In the following we summarize the reasons of these decisions.

- OWL2 is a logic-based description language for the semantic web that is able to explicitly represent complex knowledge about things and their relations. The expressiveness of OWL2 to represent machine-interpretable content overcomes other existing languages such as RDF [4], DAML [17], and DAML+OIL [6]. Several semantic reasoners exist for OWL, as Fact++ from the University of Manchester and Hermit from the University of Oxford, which allow inferring automatically semantic properties of ontology defined-classes. Furthermore, the Semantic Web Rule Language SWRL [12] is an extension of the OWL model-theoretic semantics that provides a formal meaning to OWL ontologies by including Horn-like rules written in RuleML. By this means, instance-based semantic assumptions in floor plan classes can be added to our ontology and automatically be reasoned. Finally, query languages as SPARQL [10] and OWL-SAIQL [13] allow to query the OWL ontology similarly to SQL for relational databases. OWL2, SWRL, and SPARQL are taken as W3C recommendation, which assures their promotion, maintenance, and upgrade.
- Protégé is a software developed by the University of Stanford to construct ontologies and knowledge-based applications in a friendly UI. It is currently used in several research and private projects given its wide spectrum of functionalities for ontology design and application.[1] It supports, among others, OWL, SWRL, and SPARQL.

In the following, we firstly explain the floor plan ontology and we subsequently describe how we have integrated real data into it.

2.1 Floor Plan Ontology

The design of the floor plan ontology started by deciding the functionality that it is intended to. In our case, we have constructed an ontology to represent the knowledge of floor plan documents within the scope of architectural understanding. This definition encapsulates the structural configuration of these documents, the classes (concepts), properties (attributes), and relations (contextual dependences). Despite it is worthy to remark that this is our own definition and it will vary for different applications, images, and experts, we have defined this ontology taking into account several considerations. We have contacted a team of architects to address their needs in automatic interpretation applications. We experienced several cooperations with research and private companies aiming for different applications related to floor plan interpretation. We have considered other floor plan definitions in the literature that entail some sort of structural understanding, such is the case of [21] for evacuation building simulation, and [20] for structural floor plan retrieval. Additionally, we have also been inspired

[1] http://protege.cim3.net/cgi-bin/wiki.pl?ProjectsThatUseProtege.

by the relevance of the structural information for high-level understanding in graphical documents, i.e. flowchart interpretation in patent documents [18].

Let us further describe the main elements of the ontology:

- **Class Taxonomy:** The classes in the ontology define objects or concepts. In our case, the classes are the structural concepts appearing in architectural drawings: Building, Room, Domain, Wall, Door, Window, etc. Notice that these classes are disjoint under a semantic point of view. This means that one instance can only belong to one of the defined classes, e.g a *wall* belonging to the class Wall cannot be at the same time an instance of the Room. Semantically, all these classes are disjoint siblings from a common parent class named StructuralElement, see Fig. 1. For instance, a building is a individual of the class Building, which is a *kind of* StructuralElement.

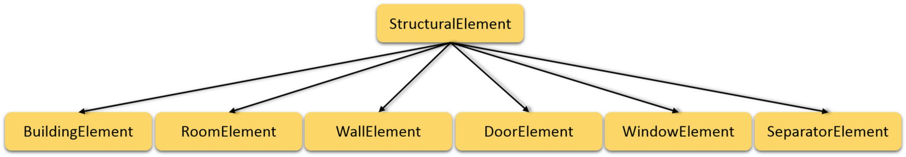


Fig. 1. Class taxonomy.

- **Object Properties:** Object properties are pairwise relations between individuals. In the floor plan ontology they describe the structural dependences that can relate two objects. For instance, rooms may be related in terms of *neighborhood* and *accessibility*, and walls, doors and windows in terms of *incidence*, see Fig. 2. Furthermore, we also define a taxonomy of object properties, e.g. the relations *hasRoom*, *hasWall*, *hasDoor*, *hasWindow*, and *hasSeparation* are subproberties of *hasStructuralElement*. This last relation is transitive, which implies that, when a individual *A hasStructuralElement B* and, at the same time, this *B hasStructuralElement C*, *A hasStructuralElement C*.
- **Data Properties:** The object classes may have defined some properties or attributes that link their individuals to an XML Schema Datatype. For instance, we defined in our syntactic representation that buildings and rooms cover an area or space in a building. Therefore, we define a data property named *hasArea* that relates the individuals of these classes with a numerical value. In Fig. 3 we show the data properties for the StucturalElements.

2.2 Introducing Real Instances into Our Knowledge Base

Once our domain is described, we have created a knowledge base by introducing real instances into the ontological definition. Our aim is to perform semantic reasoning on this data and thus, to validate our ontological design together with our incoming floor plan representations. This input data comes from two different sources. On the one hand, it is acquired from the floor plan interpretation

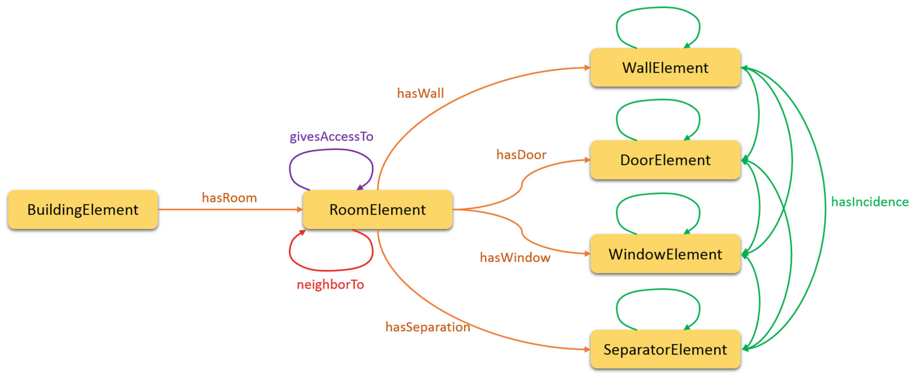


Fig. 2. Object properties.

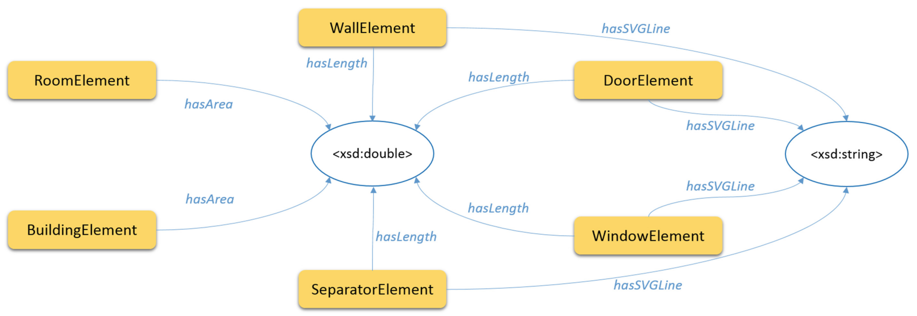


Fig. 3. Data properties.

method in [7]. This recognition approach outputs the graph representation that carries the structure of each document. On the other hand, it is collected from the structured groundtruth in [8]. These manually annotated documents not only incorporate the labellings of the objects, but they also make explicit the structural relations between objects.

Even though there are several frameworks and APIs available to transform our definition into a practicable implementations, e.g. Jena [1] and Sesame [2] in JAVA^{TM}, we have addressed this task in the opposite way. We have introduced our instances into the OWL definition and thus, used Protégé to perform the reasoning. This has been done by implementing a simple wrapper that is able to parse both, the interpreted representations and the SVG files from the groundtruth.

3 Experimental Validation

In this section we explain two use-cases for our knowledge-base of floor plans as examples of the multiple possibilities when semantic reasoning is available.

Firstly, we show how the automatic reasoning has helped us to perform further classification of our knowledge agreeing to new class definitions. Secondly, we show how this automatic classification has allowed us to do both, validate the consistency of our groundtruth and to improve a strictly bottom-up interpretation method.

3.1 Automatic Instance Classification for Groundtruth Validation

On the ontological specification presented in this paper, we have created new object classes whose individuals comply certain characteristics. Then, we use the reasoner to automatically compute the new class hierarchy and classify the instances that satisfy that specifications.

We have created a new object property namely *isPerimeterOf* that relates an architectural physical primitive –wall, door, or window– with a building instance; it specifies that a certain primitive is part of the exterior perimeter of a particular building. Then, we can define three object classes ExteriorWallElement, ExteriorDoorElement, and ExteriorWindowElement consisting of exterior primitives:

ExteriorWall := WallElement **and** (isPerimeterOf **some** BuildingElement)
ExteriorDoor := DoorElement **and** (isPerimeterOf **some** BuildingElement)
ExteriorWindow := WindowElement **and** (isPerimeterOf **some** BuildingElement).

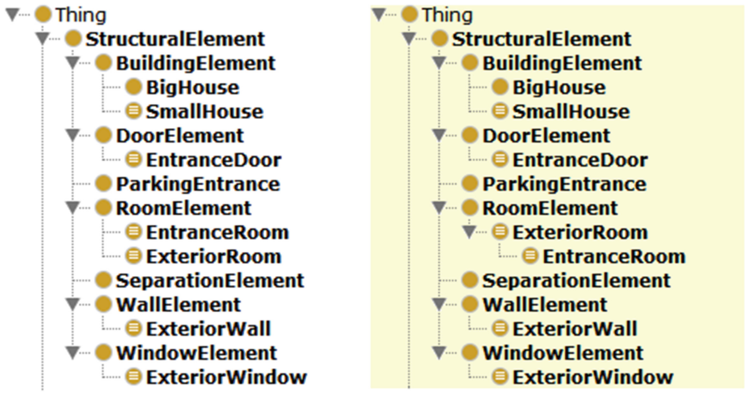
When we run the reasoner, it automatically infers that ExteriorWall, ExteriorDoor, and ExteriorWindow are actually subclasses of WallElement, DoorElement, and WindowElement respectively. Furthermore, it automatically classifies that primitive individuals with a valid *isPerimeterOf* relation. Now we want to define what an exterior room is. We can do it as follows:

ExteriorRoom := RoomElement
and ((hasWall **some** ExteriorWall)
or (hasDoor **some** ExteriorDoor)
or (hasWindow **some** ExteriorWindow)).

Therefore, an exterior room is a room instance that has a wall, a door, or a window that belongs to the exterior perimeter of a building. Let's now define what an entrance room of a building is:

EntranceRoom := RoomElement
and (**hasDoor some ExteriorDoor**).

The reader may notice that both, ExteriorRoom and EntranceRoom are defined as subclasses of RoomElement. Yet, the reasoner actually infers that the class EntranceRoom is a subclass of ExteriorRoom, i.e. all instances of EntranceRoom are instances of ExteriorRoom at the same time. Figure 4 shows a snapshot of the class hierarchy before and after applying the reasoner. This feature is really helpful when the size of the ontology (the number of classes) starts to significantly increase and keeping the semantic consistency becomes a challenging task.



```
Thing
  StructuralElement
    BuildingElement
      BigHouse
      SmallHouse
    DoorElement
      EntranceDoor
    ParkingEntrance
    RoomElement
      EntranceRoom
      ExteriorRoom
    SeparationElement
    WallElement
      ExteriorWall
    WindowElement
      ExteriorWindow
```

```
Thing
  StructuralElement
    BuildingElement
      BigHouse
      SmallHouse
    DoorElement
      EntranceDoor
    ParkingEntrance
    RoomElement
      ExteriorRoom
        EntranceRoom
    SeparationElement
    WallElement
      ExteriorWall
    WindowElement
      ExteriorWindow
```

Fig. 4. Class hierarchy before and after the automatic inference.

Now, we can imagine that this knowledge base belongs to real estate company that allows to search online their available flats for rent. It may be interesting to classify the dwellings according to their usable space. Therefore, we can predefine some classes to define different building types concerning their area:

$$\begin{aligned}&\text{Studio} := \text{BuildingElement}\\&\textbf{and}\ \text{hasArea}\ \textbf{double}[<= 20]\end{aligned}$$

$$\begin{aligned}&\text{SmallHouse} := \text{BuildingElement}\\&(\textbf{and}\ \text{hasArea}\ \textbf{double}[> 20])\\&(\textbf{and}\ \text{hasArea}\ \textbf{double}[<= 70])\end{aligned}$$

$$\begin{aligned}&\text{BigHouse} := \text{BuildingElement}\\&\textbf{and}\ \text{hasArea}\ \textbf{double}[> 70].\end{aligned}$$

We can also declare this classes using SWRL. For instance in the case of the Studio:

$$\text{BuildingElement}(?x), \text{hasArea}(?x, ?y), \text{lessThanOrEqual}(?y, 20) \rightarrow \text{Studio}(?x).$$

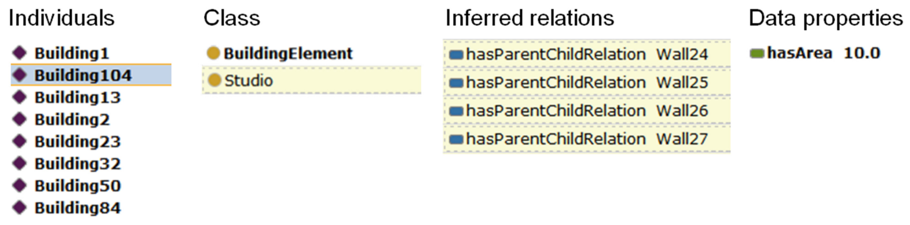
SWRL also allow us to define constrains between relationships. For instance, we can define that all the rooms that are accessible from each other are also neighbors:

$$\text{givesAccessTo}(?x, ?y) \rightarrow \text{hasNeighbor}(?x, ?y).$$

Finally, imagine that we are very interested on finding buildings that are exterior because we do like natural illumination at home. The semantic concept of exterior building can be defined as those building instances that at least have 3 rooms at the boundaries of the building. We therefore can define the ExteriorBuilding class as:

$$\text{BuildingElement}(?x), \text{hasRoom}(?x, ?y), \text{ExteriorRoom}(?y), \text{makeBag}(?b, ?y),$$
$$\text{greaterThan}(?b, 2) \rightarrow \text{ExteriorBuilding}(?x).$$

To validate the proper instance classification regarding these definitions, we have introduced some of the interpreted and groundtruth instances into our knowledge-base. Our wrapper writes into the ontology the structured data, already specifying which instances belong to the exterior boundary of a building, and the reasoner automatically performs the classification. In Fig. 5, we show a simple example to illustrate this automatic classification.



Individuals	Class	Inferred relations	Data properties
◆ Building1	● BuildingElement	hasParentChildRelation Wall24	hasArea 10.0
◆ Building104	● Studio	hasParentChildRelation Wall25	
◆ Building13		hasParentChildRelation Wall26	
◆ Building2		hasParentChildRelation Wall27	
◆ Building23			
◆ Building32			
◆ Building50			
◆ Building84			

Fig. 5. Automatic instance classification. The reasoner categorizes the instance *Building104* as a Studio according to its area. The reasoner also infers the building *parentChildRelation* with those primitives that belong to its rooms.

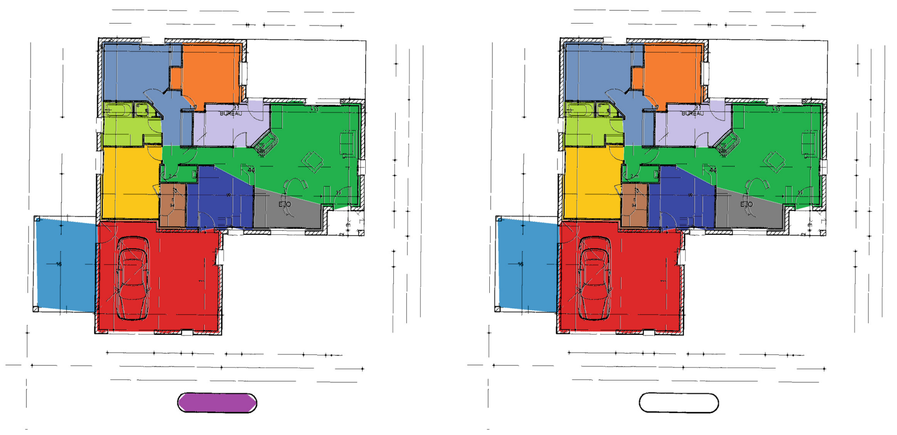
3.2 Groundtruth Validation and Automatic Interpretation Improvement

The automatic verification of the instance description w.r.t the domain ontology has been a crucial process for the generation of the consistent floor plan groundtruth, named CVC-FP, presented in [8]. We have created a labeling tool[2] that allows to make specific the structural relations between the different architectural elements. Nevertheless, this tool does not control whether the relations between instances are well defined in terms of the knowledge model. Since the manual annotation is susceptible to errors, the consistency of labeled images can be strongly harmed. Therefore, we have incorporated every groundtruthed image into our knowledge base and have used the reasoner to spot transgressing

[2] The SGT-tool and the CVC-FP database are freely available at http://dag.cvc.uab.es/resources/floorplans.

instances w.r.t the domain definition. From the 122 labeled images, 23 labeling errors have been reported. Most of them produced by a violation of the structural relations domain or scope. For instance, when trying to define as accessible a wall and a room instances. In these cases, the reasoner outputs the encountered inconsistency and facilitates the correction of the mislabeled images.

In addition to that, we have used our domain definition to add a semantic layer on top of the bottom-up interpretation presented in [7]. This layer analyzes the graph representation output by the system and allows to detect inconsistencies w.r.t the knowledge model. For instance, the domain definition states that every room in a floor plan must be neighborly connected to at least another room. Therefore, the knowledge model can parse the room connectivity graph and discard those instances that are isolated. Quantitatively, this semantic analysis improves the recognition accuracy on two of the four datasets of the CVC-FP. These database is composed of real images split into datasets according to the graphical notation of the walls: Black, Textured, Textured2, and Parallel. On the Textured dataset from 85.7% to 89.4%, and in the Textured2 dataset from 40.4% to 41.7%. Meanwhile, the results on the Black and Parallel datasets remain the unaltered. This enhancement on both textured datasets is produced by the fact that, on these collections, multiple false positive rooms are obtained at the outskirts of the building models, see an example in Fig. 6, but they are detected and ruled out by semantic reasoner.



(a) Result obtained by [7]. (b) Result obtained by [7] with subsequent semantic reasoning.

Fig. 6. Semantic reasoning impact on room segmentation.

4 Conclusions

In this paper, we have created an ontological definition of the semantic meaning expressed in floor plan documents. This ontology has allowed us to specifically define the architectural concepts, their attributes, and relations. This ontology has been written in the Ontology Web Language due to its expressibility power and its multiple available tools. Furthermore, we have created a knowledge base of floor plans by introducing real instances of conveniently annotated documents into our semantic definition. This knowledge-base has allowed us to improve the performance of a recent floor plan interpretation system and to correct the manual mislabelings on a structural database of floor plans.

References

1. Apache Jena - A free and open source Java framework for building Semantic Web and Linked Data applications (2014). https://jena.apache.org/index.html
2. OpenRDF Sesame - A de-facto standard framework for processing RDF data (2014). http://www.openrdf.org
3. Protégé 5: A free, open-source ontology editor and framework for building intelligent systems (2014). http://protege.stanford.edu
4. RDF: Resource Description Framework (2014). http://www.w3.org/RDF/
5. Bhatt, M., Hois, J., Kutz, O.: Ontological modelling of form and function for architectural design. Appl. Ontol. **7**(3), 233–267 (2012)
6. Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: DAML+OIL: Reference Description (2001). http://www.w3.org/TR/daml+oil-reference
7. de las Heras, L.-P., Ahmed, S., Liwicki, M., Valveny, E., Sánchez, G.: Statistical segmentation and structural recognition for floor plan interpretation. Int. J. Doc. Anal. Recogn. (IJDAR) **17**(3), 221–237 (2014)
8. de las Heras, L.-P., Terrades, O.R., Robles, S., Sánchez, G.: CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool. Int. J. Doc. Anal. Recogn. (IJDAR) **18**(1), 15–30 (2015)
9. Gruber, T.R.: A translation approach to portable ontology specification. Knowl. Acquisition **5**, 199–220 (1993)
10. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language (2013). http://www.w3.org/TR/sparql11-query/
11. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2: Web Ontology Language (2012). http://www.w3.org/TR/owl2-primer
12. Horrocks, I., Patel-Schneider, P.F.., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a semantic web rule language combining OWL and RuleML (2004). http://www.w3.org/Submission/SWRL
13. Kubias, A., Schenk, S., Staab, S., Pan, J.Z.: OWL SAIQL - an OWL DL query language for ontology extraction. In: Proceedings of the International Workshop on OWL: Experiences and Directions (2007)
14. Maillot, N.E., Thonnat, M.: Ontology based complex object recognition. Image Vis. Comput. **26**(1), 102–113 (2008)
15. Möller, M.: Fusion of spatial information models with formal ontologies in the medical domain, Ph.D. thesis, German Research Center for Artificial Intelligence (DFKI) (2011)
16. Nwogu, I., Zhou, Y., Brown, C.: An ontology for generating descriptions about natural outdoor scenes. In: IEEE International Conference on Computer Vision Workshops, pp. 656–663 (2011)
17. Pagels, M.: DAML - The DARPA Agent Markup Language (2006). www.daml.org

18. Piroi, F., Lupu, M., Hanbury, A., Sexton, A., Magdy, W., Filippov, I.: Clef-ip: retrieval experiments in the intellectual property domain. In: CLEF Evaluation Labs and Workshop (2012). No. Online Working Notes
19. Tongphu, S., Suntisrivaraporn, B., Uyyanonvara, B., Dailey, M.N.: Ontology-based object recognition of car sides. In: International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp. 1–4 (2012)
20. Weber, M., Langenhan, C., Roth-Berghofer, T., Liwicki, M., Dengel, A., Petzold, F.: a.SCatch: semantic structure for architectural floor plan retrieval. In: Proceedings of the International Conference on Case-Based Reasoning, pp. 510–524 (2010)
21. Zhi, G.S., Lo, S.M., Fang, Z.: A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. Comput. Aided Des. **35**(1), 1–14 (2003)