Building Knowledge about Buildings

Matthew T. Young and Eyal Amir

University of Illinois, Urbana-Champaign
3314 Siebel Center
201 N Goodwin Ave, Urbana, IL 61801
mtyoung@uiuc.edu, eyal@cs.uiuc.edu

Abstract

The ability to encode information about the structure of buildings is essential for the development of applications which are able to reason about buildings and answer queries concerning their design and function. Currently, the only methods of encoding structural information are bloated, inconsistent, and poorly defined, making the development of these applications impossible. In this paper, we outline a new language for encoding this knowledge using description logic. This language is designed to be both simplistic and powerfully expressive.

Introduction

Buildings represent a unique challenge for knowledge engineers. Encoding all the information about a single building involves not only describing the materials which compose it, but also the ways these materials are combined, along with higher-level architectural details. CAD programs typically represent buildings purely as geometric objects, with no information about the underlying structure (such as "this is a wall"). The IAI has developer the IFC building model, but this model is incredibly complex and meant to be used for the creation of blueprints and details for the actual construction of a building. This is a much greater level of detail than most applications require.

A simple representation language for buildings allows knowledge engineers to quickly encode information, and allows automated reasoners to quickly extract this information in order to answer queries. Also, a simple model enables content providers to more easily share information, allowing the creation of very powerful searching, aggregation, and classification tools.

We have developed a simple model by stripping away all of the material details from the representation. This model purely represents architectural features such as doors, rooms, staircases, etc, and how they are connected to each other. While the model does allow the user to describe what something is made of, this is in a purely semantic sense and carries no explicit restrictions. This simplification allows the model to be extremely expressive, encompassing the entire domain of buildings, without the need for domain experts and hours of time to enter data.

The Building Language

We wish to specify the building representation language in such a way that it stores the structure of the building compactly and answer queries quickly. We also want to ensure that each building is physically possible and that all buildings are encoded consistently. Description logic is a natural choice for this task, as it has been used to build ontologies for representing knowledge for many years.

The specific description logic language we chose to use is OWL. OWL was chosen because it is being used for the Semantic Web, and there are many useful web-based applications for a building representation language (such as a house-hunting website). The flavor of OWL we are using is OWL DL, which is computationally complete and decidable (Smith, 2004), and therefore should give good performance.

The hierarchical bulleted list below (Figure 1) defines all the classes in the building language. Indented bullets are subclasses of the class above them. The classes are described in more detail in the next section.

- Building
- External_Feature
  - Exterior_Wall
  - Roof
  - Roof_Fixture
  - Wall_Feature
    - Door
    - Window
- Internal_Feature
  - Floor
  - Fixture
  - Interior_Wall
  - Room
  - Stairwell
- Material
  - Aesthetic_Material
  - Complex_Structure
  - Construction_Material

Figure 1 – List of Classes

Description of Classes in Building Language

Each class contains many properties describing the features of that class. In this paper, properties are written as *Property_Name* (number Type). For example, to say that the property named Has_Internal_Features can filled by a list of multiple instances of the type Internal_Feature, it is

written *Has_Internal_Features* (multiple Internal_Feature). The "number" component can either be "single" (functional), or "multiple", (a list).

As can be seen from Figure 1, there are four top-level classes in the building language. Each of these classes and the subclasses they contain is described in a separate subsection below.

Building: Serves to connect all features together into a single unit. It has two properties: *Has_Internal_Features* (multiple Internal_Feature) and *Has_External_Features* (multiple External_Feature), that identify the internal and external features that are ascribed to this building.

External_Feature: Includes building elements that are a part of the exterior of a building. It has one property: *Ext_In_Building* (single Building), that identifies the building containing this feature. Its subclasses are particular types of external features. "Exterior_Wall" is a wall with at least one face on the outside of the building. "Roof" is a roof on a building, which could be an overhang. "Roof_Fixture" is a fixture that is found on a roof, such as a satellite dish, chimney, etc. "Wall_Feature" is a feature that can be found in a wall; it has two subclasses: "Door" and "Window". The properties of all these subclasses are detailed in Table 2 (For brevity, 'Construction' is abbreviated 'C').

Class Name	Properties
Exterior_Wall	*Attached_To* (multiple Interior_Wall, Exterior_Wall): The walls sharing a joint with this wall.
	C_Made_Of (multiple C_Material): The materials that make up this wall.
	Has_Wall_Features (multiple Wall_Feature): The features (windows and doors) in this wall.
Roof	*C_Made_Of* (multiple C_Material): The materials that make up this roof.
	Has_Roof_Fixtures (multiple Roof_Fixture): All objects attached to the roof.
	Spans_Walls(multiple Exterior_Wall): The walls to which this roof is attached.
Roof_Fixture	*General_Made_Of* (multiple Material): The materials or structures that make up this roof fixture.
	On_Roof (single Roof): The roof on which this fixture is located.
Wall_Feature	*C_Made_Of* (multiple C_Material): The materials that make up this wall feature.
	In_Wall (multiple Exterior_Wall): The wall in which this feature is embedded.

Table 2 – Properties of External_Feature Subclasses

Internal_Feature: Includes building elements which are contained entirely within the building's interior. It has one property: *Int_In_Building* (single Building), that identifies the building containing this feature. Its subclasses are particular types of internal features. "Fixture" is a permanent or semi-permanent user of floor space, such as a bathtub, counter, or large piece of furniture. "Floor" is a level of the building, typically a surface meant to be walked on. "Interior_Wall" is a wall contained entirely within a building. "Room" is a room in a building; it may be completely compartmentalized or a division of a larger area. "Stairwell" is a connection between multiple floors; in spite of the name a Stairwell may be a set of stairs, elevator, or escalator. The properties of all these subclasses are detailed in Table 3 (For brevity, 'Construction' is abbreviated 'C').

Class Name	Properties
Fixture	*General_Made_Of* (multiple Material): The materials or structures that make up this fixture.
	In_Room (single Room): The room containing this fixture.
Floor	*Above_Floor* (single Floor): The floor this floor is above. It may be non-existent.
	Below_Floor (single Floor): The floor this floor is below. It may be non-existent.
	C_Made_Of (multiple C_Material): The materials that make up this floor.
	Has_Interior_Walls (multiple Interior_Wall): The interior walls that exist on this floor.
	Has_Rooms (multiple Room): The rooms whose floor is this floor. Some rooms may have empty space continuing upward for several floors, but they are not considered to be on those floors.
Interior_Wall	*Attached_To* (multiple Interior_Wall, Exterior_Wall): The walls sharing a joint with this wall.
	C_Made_Of (multiple C_Material): The materials that make up this wall.
	Wall_On_Floor (multiple Floor): The floors on which this wall is present.
Room	*Bounding_Walls* (multiple Interior_Wall, Exterior_Wall): The walls which define the boundaries of this room.
	Ceiling (single Floor, Roof): The ceiling for this room. May be non-existent.

	Has_Exits (multiple Door): The exits from this room to the outside of the house. Note the difference between this property and Shares_Doorway_With. *Has_Fixtures* (multiple Fixture): The fixtures located in this room. *Has_Windows* (multiple Window): The windows for this room. *Room_On_Floor* (multiple Floor): The floors that this room is located on. Note that a room can have multiple floors, such as a theater which has a seating area and a stage. *Shares_Doorway_With* (multiple Room): The other rooms which share a doorway with this room.
Stairwell	*Connects_Floors* (multiple Floor): The floors connected by this stairwell. *Connects_Rooms* (multiple Room): The rooms connected by this stairwell.

Table 3 – Properties of Internal_Feature Subclass

Material: Includes the substances and structures that the various features of the building are made of. It has no properties. This is because Materials are only used to add richness to the description of a building, not for reasoning about physical properties. Its subclasses are specific sets of substances or structures. "Aesthetic_Material" is a material which is used to make objects more appealing to humans, such as cloth, or paper. "Complex_Structure" is a structure that cannot be represented as a few materials, such as a dishwasher or AC unit. "Construction_Material" is a material which is generally used to define the structure of something, such as wood, concrete, drywall, or glass.

Language Constraints

The classes and properties define the structure of the building representation language. Constraints ensure that a user can only encode buildings that are consistent with the rules of spatial geometry and the general principles of building design.

Constraints in description logic are asserted conditions in the form of logical statements. For example, ∃*Has_External_Feature* (Door) means the property *Has_External_Feature* must contain at least one Door in its fillers. Each of the top-level classes and the subclasses they contain is described in a separate subsection below (For brevity, 'Construction' is abbreviated 'C').

Building: ∃*Has_External_Feature* (Door, Roof, Exterior_Wall), ∃*Has_Internal_Feature* (Floor, Room).

External_Feature: ∃*Ext_In_Building* (Building).

- Exterior_Wall: ∃*C_Made_Of* (C_Material).
- Roof: ∃*C_Made_Of* (C_Material), ∃*Spans_Walls* (Wall).
- Roof_Fixture: ∃*General_Made_Of* (Material), ∃*On_Roof* (Roof).
- Wall_Feature: ∃*C_Made_Of* (C_Material), ∃*In_Wall* (Exterior_Wall).

Internal_Feature: ∃*Int_In_Building* (Building).

- Fixture: ∃*General_Made_Of* (Material), ∃*In_Room* (Room).
- Floor: ∃*C_Made_Of* (C_Material), ∃*Has_Rooms* (Room).
- Interior_Wall: ∃*C_Made_Of* (C_Material), ∃*Wall_On_Floor* (Floor).
- Room: ∃*Room_On_Floor* (Floor), ∃[*Has_Exits* (Door) ∪ *Shares_Doorway_With* (Room)|, ∃*Bounding_Walls* (Inter_Wall ∪ Exter_Wall).
- Stairwell: *Connects_Floors* ≥ 2 (Floors), *Connects_Rooms* ≥ 2 (Rooms).

Material: No asserted conditions.

Constraints that Cannot be Encoded

Several more complicated constraints would be useful to include in the building representation language, but simply cannot be encoded due to the limitations of OWL. A couple of examples are "A Floor cannot have an Above_Floor which is above its Below_Floor" and "A Stairwell cannot have more than one Room on the same Floor in Connects_Rooms." The reason these and many other rules cannot be encoded is that no constraint can inspect the actual values of any property. Constraints can only enforce type and cardinality.

Example of How to Enter Data: Simple House

To demonstrate how easy it is to build knowledge using the building language, this section contains a set of steps detailing how to encode a Simple Two Story (STS) house. Figure 4 shows a simplified blueprint of this house, with one possible labeling for the walls (note that interior and exterior walls are labeled separately). Solid lines are walls (except for the stairwell) and dotted lines are doorways.

1. Create a "Building" named **STS_House**.
2. Create four "Exterior_Walls" named **Ext_Wall1** through **Ext_Wall4**. Set *Ext_In_Building* for each to STS_House.
3. Create two "Doors" named **Front_Door** and **Back_Door**. Set *Ext_In_Building* for each to STS_House. Set *In_Wall* for Front_Door to Ext_Wall3 and *In_Wall* for Back_Door to Ext_Wall1.
4. Create a "Window" for each window on the outside of the house. There could be many windows, so they are not all individually detailed here. For each, set *Ext_In_Building* to STS_House and *In_Wall* to the appropriate Ext_Wall.
5. Create a "Roof" named **STS_Roof**. Set *Ext_In_Building* to STS_House and *Spans_Walls* to

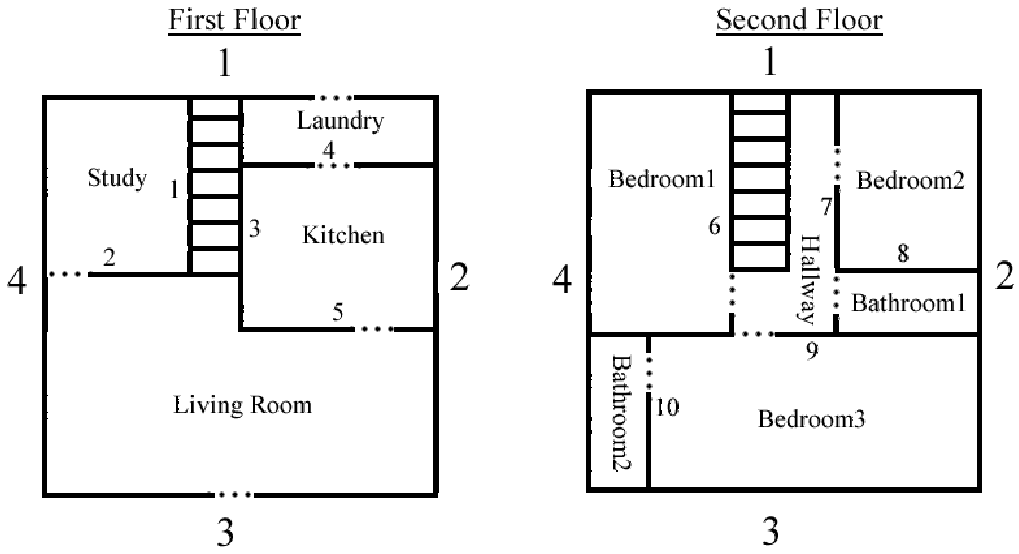

Figure 4 - Simple Two Story House

Ext_Wall1, Ext_Wall2, Ext_Wall3, and Ext_Wall4.
6. Create a "Roof_Fixture" named **AC_Unit**. Set *Ext_In_Building* to STS_House, *On_Roof* to STS_Roof.
7. Create two "Floors" named **First_Floor** and **Second_Floor**. Set their *Int_In_Building* to STS_House.
8. Set *Below_Floor* for First_Floor to Second_Floor and set *Above_Floor* for Second_Floor to First_Floor.
9. Create ten "Interior_Walls" named **Int_Wall1** through **Int_Wall10**. Set *Int_In_Building* for each to STS_House. Set *Wall_On_Floor* for Int_Wall1 through Int_Wall5 to First_Floor and set *Wall_On_Floor* for Int_Wall6 through Int_Wall10 to Second_Floor.
10. Fill in the *Attached_To* lists for Int_Wall1-Int_Wall10 and Ext_Wall1-Ext_Wall4, using Figure 4 for reference. For example, Int_Wall1 should have (Ext_Wall1, Int_Wall2).
11. Create ten "Rooms" named as shown in Figure 4. Set *Int_In_Building* for each Room to STS_House. Set *Room_On_Floor* for Study, Living_Room, Kitchen, and Laundry_Room to First_Floor and set *Room_On_Floor* for all the other rooms to Second_Floor. Set *Ceiling* for Study, Living_Room, Kitchen, and Laundry_Room to Second_Floor and set *Ceiling* for all the other rooms to STS_Roof. Set *Has_Exits* for Living_Room to Front_Door and for Laundry_Room to Back_Door. Set *Has_Windows* for each Room to the appropriate list of Windows.
12. Fill in the *Bounding_Walls* lists for all Rooms with the appropriate Int_Wall1-Int_Wall10 and Ext_Wall1-Ext_Wall4. For example, Study should have (Ext_Wall1, Ext_Wall4, Int_Wall1, Int_Wall2).
13. Fill in the *Shares_Doorway_With* lists for all Rooms with the appropriate other Rooms. For example, Living_Room should have (Study, Kitchen).
14. Create a "Stairwell" named **Main_Stairwell**. Set *Int_In_Building* to STS_House.
15. Fill in *Connects_Floors* for Main_Stairwell with First_Floor and Second_Floor. Fill in *Connects_Rooms* for Main_Stairwell with Living_Room and Hallway.
16. Create a "Fixture" for each permanent fixture or large piece of furniture in the building. There may be many of them, so they are not detailed individually here. Set *Int_In_Building* for each Fixture to STS_House. Set *In_Room* for each Fixture to the appropriate Room.
17. Create five "Construction_Materials" named **Brick**, **Drywall**, **Glass**, **Shingles**, and **Wood**. Create two "Aesthetic_Materials" named **Cloth** and **Foam**. Create a "Complex_Structure" named **Rooftop_AC_Unit**.
18. Fill in the *Construction_Made_Of* lists for all Interior_Wall, Exterior_Wall, Roof, Floor, and Wall_Feature instances with the appropriate combinations of Brick, Drywall, Glass, Shingles, and Wood.
19. Fill in the *General_Made_Of* lists for all Fixture and Roof_Relief_Feature instances with the appropriate combinations of Brick, Drywall, Glass, Shingles, Wood, Cloth, Foam, and Rooftop_AC_Unit.

References

[1] Cuenca-Grau, Bernardo; Parsia, Bijan; Sirin, Evren; Kalyanpur, Aditya. Modularity and Web Ontologies. International Conference on Knowledge Representation and Reasoning, 2006.

[2] Haarslev, V.; Möller, R.; Wessel, M. Description Logic Inference Technology: Lessions Learned in the Trenches. In I. Horrocks, U. Sattler, and F. Wolter, editors, Proc. International Workshop on Description Logics, 2005.

[3] Noy, Natalya F.; McGuinness, Deborah L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, Mar. 2001.

[4] Smith, Michael K.; Welty, Chris; McGuinness, Deborah L. 2004. OWL Web Ontology Language Guide. W3 Consortium. http://www.w3.org/TR/owl-guide/