Cody Lanier
Cody Prior

CSc-180
Spring 2014
Project 1 - EXPERT SYSTEMS

Project Overview

Our expert system is designed to discern between many different commonly mistaken European architectures from about 600 BC to the 17th century. It knows about Greek, Roman, Romanesque, Renaissance, and Gothic European styles. The system will prompt the user for various inside and outside features of the building, as well as the date that the building was approximate constructed. We wanted a user to be able to query the system based on an image of a building or using a personal description of a building, and for the expert system to be able to correctly guess the style in which that building was constructed.

The Expert

Our knowledge expert is Lexi Prior. She is a senior architecture student at Cal Poly. She has studied architecture for 5 years and graduates in June. She spent 3 semesters studying architecture history and gained a comprehensive understanding of architecture techniques and styles.

Scope

When we initially approached Lexi for a potential project, we had the idea that we would create an expert system that could differentiate between a large set of modern architectural styles. This idea was very quickly shot down for two reasons. One was that the scope of the project was deemed too big for the limited time frame we had and the other was that our expert told us that most architectural styles were different enough that our system would be able to quickly eliminate a large set of possibilities. Our expert later came to us with an alternative idea: a system that would differentiate between 5 specific European styles. Our expert told us that they had enough similarities that most people would commonly mistake them. The project's scope was much more reasonable and the system appeared to be much more useful, so we took on the project.

Knowledge Engineering

The first time we met with Lexi, we had her regurgitate as many facts about the several commonly mixed up European architectural types as she could. We compiled these facts into several bulleted lists to make them easy to go through. On our own time, we went over the facts and compiled a list of questions to ask the expert during our next meeting with her. When we next met with Lexi, we asked her to try and group like facts and we ended up narrowing our fact base a little bit so we could really focus on the parts that would help our program make decisions. Lexi had prepared a chart for us, which compared the differences between the buildings' similarities, like columns, for example. That chart would be integral to our expert system. The final meeting with our expert was to ask additional questions that came up during the actual programming of the system. We also gathered some information from her to prepare this document, at that final meeting.

We did not constrain the application much. We felt that it would be best to use just about everything that was discussed in the first two meetings so that our system would be able to make good decisions. The fuzzy logic was the hardest for us to come up with. Originally, we thought we would use the dates of the buildings somehow, but ended up using descriptions of the columns to fuzzify data. We had to convolute the question about how decorated the column capitals were, in order to have some data that we could fuzzify. We made it a decoration scale from 1 to 10 with 10 being the most decorated, instead of a simple question asking if the columns were plain, had swirls, or depicted leaves and flowers. We also left Corinthian style columns out of our fuzzy logic because they are very similar to composite and were shared by all 5 styles.

Expert System Design

We started grouping the information given to us by the expert by European style. We got as many facts as possible about each type of building that she determined were commonly mistaken architectural types. We then narrowed this down into common characteristics of the different buildings, as well as stand out characteristics. This allowed us to start to develop a question tree, which would ultimately lead the user down a path towards an answer. The system works by prompting the user for characteristics about the building starting from the outside and works its way indoors, assuming those portions of the building are available to the user. If the user can only see the inside of the building then the system only asks about inside features like arches and ceiling shapes. If the user can only see the outside, The system will ask about outside features such as whether there are visible columns or if the building has buttresses or towers. As the user continues to feed the system information about the building, the expert system checks the building facts against a list of facts about each architectural style. If the building has a feature not incorporated into a specific style, that style is removed from the list of possible final answers. The system then tailors the remaining questions around trying to differentiate between the surviving styles. Once the system has collected all the information that it could, it will output to the user its best guess as to what the style could be or is.

Fact Overview:

Style facts correspond to potential styles for the building (Greek, Roman, Romanesque, Gothic, and Renaissance). These facts are all asserted at the start of the program and are retracted as the system narrows down the possible styles of the building based on input.

Feature facts are simple yes/no facts as to whether or not the building being identified has a specific feature.

Column facts are unique in regard to feature facts in that we actually specify the type of column based on information from the user rather than simply prompt for yes or no questions for each type. The four types of columns are doric, ionic, composite, and tuscan.

Rules Overview:

Question rules are used to elicit input from the user.

Negation rules are used to eliminate possible styles.

Goal rules are used to output either the guaranteed style or a list of possible styles at the end of execution.

Inputs

View: The choice of being able to see either just the outside, just the inside, or both parts of the building.

Feature inputs: Yes or no responses to questions related to building features.

Column details: numeric inputs for the width, height, and opinion of how decorated the column is.

Outputs

The system has 4 possibilities in terms of output:

1. System finds a single match at which point it outputs “The building style is (The style)”.

2. System finds multiple matches and outputs a list with “Possible Style: (a style)”.

3. System is unable to reduce the list of possible styles (This will happen if the user sees none of the asked about features) and outputs "Unable to make a determination based on current information, all styles are valid" along with a list of all the styles.

4. System outputs nothing, this is a specific issue related to the column section and is explained in the *Known Issues and Bugs* section of this document.

Fuzzy Logic Overview

We are using fuzzy logic during the column identification process in the expert system. When the program begins, it will ask the user a series of questions which can lead to an answer. If the user gets to the columns questions, the expert system will ask the user to estimate the height of the visible columns followed by a question about the approximate width of the columns. These two values are computed into a height to width ratio, so that the system may determine if the columns are thin, medium, or thick. That defines the first fuzzy set. The next question posed to the user, will ask the user to rank how decorated the capitals of the columns are. The system tells the user to input a value ranging from 1 to 10, with the value 1 representing a capital with no decoration and the value 10 representing the most decorated capital ever seen. This opinionated value is then used in the second fuzzy set, which determines if a column capital is plain, has swirls, or depicts leaves and flowers. These sets make up the FAM, and after defuzzifying the data through the third fuzzy set, which associates a value with each different type of known column, a guess as to which column the user sees is given as output. There is a catch all, unknown column, in case the defuzzification process spits out a number greater than what we anticipate.

Fuzzy Sets

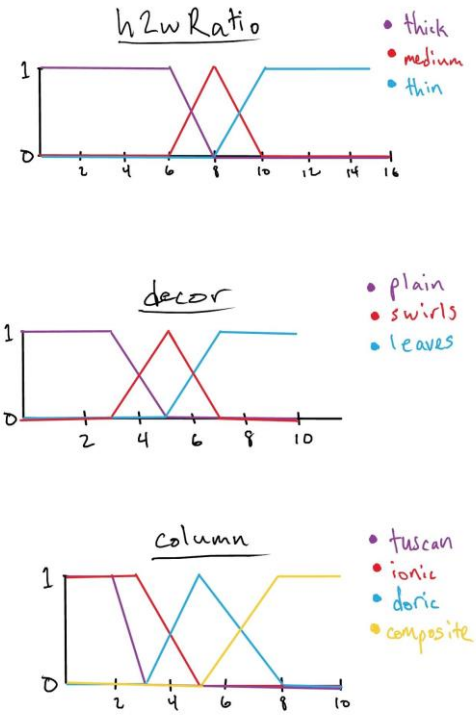
The first fuzzy set is referred to as the height to width ratio of the colum. A ratio equal to or less than 6:1 is denoted as a thick column. Thick columns trail off at the ratio of 8:1. Medium sized columns have a zero value at the 6:1 ratio and also at the 10:1 ratio. These medium columns have a value of one at the 8:1 ratio mark. Thin columns are defined as having a zero value at 8:1 and a one value at any ratio 10:1 or greater. This portion of the system is loosely based on information from our expert. The columns we are checking for do indeed have differences in size (ionic is smaller than doric for example), but we have no real hard data on the average proportions so we made them up in order to incorporate fuzzy logic.

The second fuzzy set is the capital decorations. Plain capitals have a one value for anything on the 1-10 scale that is 3 or less. It gets a zero value at 5. Swirls are defined as having zero values at 3 and 7, but they have their one value right in the middle at 5. Leaves and flowers have a zero value at exactly 5 and a one value for anything 7 or more. Obviously this is an inaccurate way of determining how the column is decorated and our expert did not tell us to use their information like this, but we needed to incorporate fuzzy logic and a FAM somehow into the program so we deviated from our expert.

The third fuzzy set determines the program's decision on which type of column the user is seeing and its values can range from 0 to 10. Tuscan style columns are usually thin and

have very plain capitals; therefore, they are defined as any value 2 or less and never any value 3 or more. Ionic columns also tend to be thin and plain, but more sturdy that the Tuscan style. Their values are one for anything 3 or less, trailing to zero at 5. Doric columns tend to medium or thick, with swirls around their capitals, so their values are zero at 3 and 8 and one right near the middle of that range, at 5. composite columns can be of any size, but are the most decorated. The values for composite are zeros at 5 and have a one value at 8 and above.

Fuzzy Set Graphs



Examples

The following is an example of the program identifying a Romanesque building:

```
FuzzyCLIPS> (run)
Can you see the outside of the building? Inside? Both? [outside/inside/both] both
Does the building feature either buttresses or towers? [yes/no] yes
Do you see any tympanums? (see definitions) [yes/no] no
Do you see any arches? [yes/no] yes
Are the arches pointed? [yes/no]no
Do you see any ribbed vaults? (see picture) [yes/no] yes
Are there columns around? [yes/no]no
The building's style is Romanesque
```

The following is an excerpt from the column question, a.k.a. fuzzy logic, portion of the expert system:

```
What is the column's approximate height in feet? 12
What is the column's approximate width in inches? 16
On a scale from 1 to 10, with 10 being super decorated, how decorated are the column capitals? 7
It's a composite column.
```

Conclusion

Our approach worked very well in respect to the knowledge engineering process. All of our meetings with the expert were extremely successful and built upon the previous ones, which put us closer to our overall goals. We struggled to come up with a place to put fuzzy logic in this type of expert system and that would be the major part of the engineering that we would have liked to have performed better in. This expert system was a challenge, but ultimately successful.

Known Bugs and Issues

The program can give answers related to columns that will lead to the program finding no suitable matches. Unlike the other sections which are carefully asked based on previous input, column questions are always asked if the user sees columns and the user inputs no to the clustered column question or if Gothic has already been eliminated. An example would be if all styles except for Greek are eliminated and the user enters the details of a composite column (which aren't used in Greek architecture). This won't cause problems if the user gives the program accurate information. In a more positive light, the user might actually be describing a style not covered by the program and in that case, the lack of output would be correct, though a bit jarring.

The program does not validate user input. If the user deviates in anyway from the expectations of the system, the program may become erratic and the output may or may not be accurate.

Limitations

The program only takes into account structural features of the building like columns and buttresses, which makes some styles with very distinct features, like Gothic, easy to differentiate from. Other styles, like Roman and Renaissance, are much harder to differentiate between with this limited scope. If we wanted to expand on the program, we could take into account other differences like floor plan or the general design of the building to further differentiate the styles.

APPENDIX A: INSTALLATION GUIDE

1. Open up the CLIPS executable
2. Goto File -> Load Constructs and open up archstyles.clp (This version doesn't feature fuzzy logic but is more true to the expert's knowledge) or archstylesFuzzy.clp (features fuzzy logic for the column section)
3. The rules will loaded into CLIPS and, if the file is not corrupted, the message TRUE will be displayed at the end.
4. Goto Execution -> Reset
5. Goto Execution -> Run. The program is now running.
6. If you would like to try running the program again after getting a result, repeat steps 4 and 5.

APPENDIX B: USER'S GUIDE

As soon as the user runs the program, a text prompt will appear asking what parts of the building they can see. The user must enter either "outside", "inside", or "both". Once the user inputs their response, they will be asked a series of yes/no questions about specific details of the building. User input is case sensitive and any deviation from expected input will result in erratic behavior from the program. Because certain features are difficult to accurately describe in words, a folder of visual aids has been provided for the user. Users may minimize the program and view the pictures if they need help figuring out what the program is asking for. Once the program runs out of questions, a result or list of possible styles will be output.

APPENDIX C: DEFINITIONS

Buttresses: Angled supports, often providing decoration as well as structure stability.

Capital: The cap, or top of a column or pillar.

Tympanum: Decorations over an entryway, usually depicting a scene or story.

REFERENCES

Lexi Prior

Joseph C. Giarratano. "Clips User's Guide." *Sourceforge.net.* n.p., 12/31/2007. Web. 3/4/2014. <http://clipsrules.sourceforge.net/documentation/v630/ug.pdf>.