Abstract

Building code requirements, which all buildings are subject to, are represented in building code manuals. As part of the construction process, one must hire a building-code inspector to examine the building and make sure it is in line with the codes. A large goal of this research was to figure out how to represent the Ohio building code manual in the form of expert system rules. Specifically, I dealt with representing a small subsection of the code using a logic-based programming language called Answer Set Programming. My main goal was to figure out how to represent questions in Answer Set Programming, as well as the optimal way in which to ask questions in order to determine building type. My program generates multiple answer sets until eventually the user has answered a sufficient number of questions, and a single answer set remains. The elimination of all answer sets except one is an indicator that the building type has been determined.

Overview

Why do this?

• Protection against “over-building”
• Elimination of mistakes
• Clarity on distinct boundaries associated with requirements

How were we able to start working immediately?

• This is based on Professor Skon’s marriage counselling AI research project
• Used prior “dependency” method upon starting this project

What are the main aspects of this project?

• Understanding the basic framework behind how these building codes are written and whether there is a certain order to the codes
• Trying to understand the tools necessary in order to first proceed toward a solution
• Continually learning and leveraging improved implementation

Example Code

SECTION 303
ASSEMBLY GROUP A

303.1 Assembly Group A. Assembly Group A occupancy includes, among others, the use of a building or structure, or a portion thereof, for the gathering together of persons for purposes such as civic, social or religious functions, recreation, food or drink consumption or awaiting transportation. A room or space used for assembly purposes by less than 50 persons and accessory to another occupancy shall be included as a part of that occupancy. Assembly areas with less than 750 square feet (69.7 m²) and which are accessory to another occupancy according to Section 302.2.1 are not assembly occupancies. Assembly occupancies which are accessory to Group B in accordance with Section 303.2 are not considered assembly occupancies. Religious educational rooms and religious auditoriums which are accessory to churches in accordance with Section 302.2 and which have occupant loads of less than 100 shall be classified as A-3.

Application

Answer Set Programming (ASP) is a declarative language oriented toward search problems. The language follows a non-monotonic logic and is guided by the three following informal principles:

1) Satisfy the rule of the program and believe in the head of a rule if you believe in its body
2) Do not believe in contradictions
3) Believe nothing you are forced to believe

• ASP is a collection of statements describing objects of a domain and relations between them.
• Its semantics defines the notion of an answer set — a possible set of beliefs of an agent associated with the program.
• The valid consequences of the program are the statements that are true in all such sets of beliefs.

Some of the basic lingo that are essential to writing a program in ASP:

Terms – Terms are either variables or object constants.

Predicate – Confirmation for a particular property between objects or relations between objects or relations between objects.

Atomic Statement – An expression of the form p(t₁, \ldots, tₙ) where p is a predicate symbol of arity n and t₁, \ldots, tₙ are terms. Atomic statements are also referred to as atoms.

Literal – An atom p(t₁, \ldots, tₙ) or its negation –p(t₁, \ldots, tₙ). This negation is read as p(t₁, \ldots, tₙ) is false. When referencing a literal, one simply denotes it as l.

Not – This is a logical connective referred to as default negation. The expression not l means that it is not believed that l is true. However, this does not imply that l is believed to be false. This allows the reasoner to have the option for a statement to be neither true nor false.

Rules – A rule is of the form a₀ ← a₁, \ldots, aₙ. The left-hand side of a rule is called the head whereas the right-hand side is called the body. The head of a rule is satisfied and represented as true only when every atom in the body is evaluated as true.

Constraints – These refer to rules with empty heads. Two literals that are a part of the same constraint cannot be in the same answer set.

Facts – Rules with empty bodies are known as facts.

ASP is quite versatile with applications various fields such as probability, graph theory, and phylogenetic systematics.

Conclusions

Many of the research questions we had prior to the summer were addressed as the project progressed. Each individual on the research team tried to answer slightly different questions, which allowed us to examine this project from many different angles. My main research question, that I had at the beginning of the summer, which had to do with the feasibility of implementing ASP, ended up splitting off into smaller sub-questions that fueled the learning process. The research question was answered in a sense; however, there are still many sub-questions and challenges that came about intermittently that still remain unanswered. Yes, Answer-Set Programming can be utilized to logically represent a sub-section of the content in Chapter 3. However, we realized that this code needs to be optimized as well as scaled to fit the needs of the other chapters and sections of the building code. A more thorough study of Answer-Set Programming and its capabilities will need to be investigated in order to address these further sub-questions.

Future Research

• Investigation into techniques that will optimize the ASP such that more “helpful” answer sets are brought to the screen instead of randomly generated ones
• Further research into how ASP can be applied to other chapters and section of the building code in regard to gaining insight into the scalability of ASP
• Continued conversation with more professionals in fields in or related to architecture

References


Acknowledgments

I would like to thank Professor Skon for providing guidance and wisdom throughout the process. Also, I would like to thank architectural engineer, Lance DeBenedictis, for helping us understand how these building codes are structured, which made it much easier to define goals and plans of action. Lastly, I would like to thank my research team: Joseph Schutz, Nicholas Ogilvie-Thompson, and Spalding Vance.

This work was funded by the Kenyon College Summer Science Scholars program.