# The ARC Project: Creating logical models of Gothic cathedrals using natural language processing

**Charles Hollingsworth**
Inst. for Artificial Intelligence
The University of Georgia
Athens, GA 30602
cholling@uga.edu

**Stefaan Van Liefferinge**
**Rebecca A. Smith**
Lamar Dodd School of Art
The University of Georgia
Athens, GA 30602

**Michael A. Covington**
**Walter D. Potter**
Inst. for Artificial Intelligence
The University of Georgia
Athens, GA 30602

## Abstract

The ARC project (for **A**rchitecture **R**epresented **C**omputationally) is an attempt to reproduce in computer form the architectural historian's mental model of the Gothic cathedral. This model includes the background information necessary to understand a natural language architectural description. Our first task is to formalize the description of Gothic cathedrals in a logical language, and provide a means for translating into this language from natural language. Such a system could then be used by architectural historians and others to facilitate the task of gathering and using information from architectural descriptions. We believe the ARC Project will represent an important contribution to the preservation of cultural heritage, because it will offer a logical framework for understanding the description of landmark monuments of the past. This paper presents an outline of our plan for the ARC system, and examines some of the issues we face in implementing it.

## 1 Introduction

The ARC project is designed to assist architectural historians and others with the task of gathering and using information from architectural descriptions.[1] The architectural historian is confronted with an overwhelming amount of information. Even if we restrict ourselves to Gothic architecture (our primary area of interest), any given building has probably been described dozens, if not hundreds, of times. These descriptions may have been written in different time periods, using different vocabularies, and may describe the same building during different stages of construction or renovation. Descriptions may be incomplete or even contradictory. An architectural historian should be able to extract necessary information about a building without encountering anything contradictory or unclear.

To facilitate information gathering, we propose a logic-based knowledge representation for architectural descriptions. Our approach is similar to that used by Liu et al. (2010), but while their representation took the form of a set of production rules for an L-system, ours is more closely tied to the semantics of natural language. Descriptions of various cathedrals would then be translated into this representation. The resulting knowledge base would be used to give intelligent responses to queries, identify conflicts among various descriptions, and highlight relationships among features that a human reader might have missed.

## 2 Why Gothic?

In addition to being major monuments of cultural heritage, Gothic cathedrals are particularly well-suited for logical analysis. The structure of Gothic follows a logical form. Despite variations, Gothic cathedrals present a number of typical features, such as pointed arches, flying buttresses, and a plan on a Latin cross (Figure 1). The repetition of elements

[1]This research benefited from the generous support of a Digital Humanities Start-Up Level I Grant from the National Endowment for the Humanities (Grant Number HD5110110), a University of Georgia Research Foundation Grant, and from The University of Georgia President's Venture Fund.



*Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 63–68,
Portland, OR, USA, 24 June 2011. 

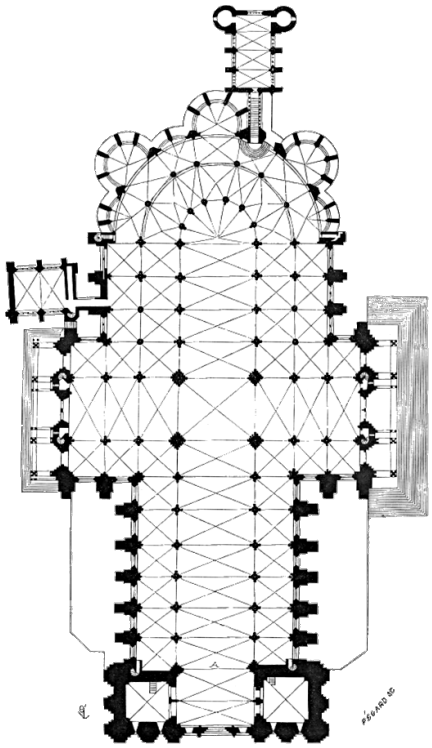

Figure 1: Example of a cathedral ground plan (Chartres, France), from Viollet-le-Duc (1854-68)

like columns and vaulting units allows for more succinct logical descriptions (Figure 2). And the historical importance of Gothic means that a wealth of detailed descriptions exist from which we can build our knowledge base.

The study of Gothic cathedrals is also important for cultural preservation. Some cathedrals have been modified or renovated over the years, and their original forms exist only in descriptions. And tragedies such as the 1976 earthquake which destroyed the cathedral in Venzone underscore the importance of architectural information. A usable and versatile architectural knowledge base would greatly facilitate the task of restoring damaged buildings.

## 3 Outline of the ARC system

The outline of the ARC system is the result of close collaboration between architectural historians and artificial intelligence researchers. While the system is still in its infancy, the complete ARC system will have three distinct modes of interaction, to be used by three different types of user. We will refer to these modes as superuser mode, administrator mode, and user mode. The superuser mode will be used to write and edit a generic model for Gothic architecture that will serve as background information prior to dealing with any specific descriptions. The administrator mode will be used to enter the details of particular buildings. The purpose of the user mode will be to allow end users to submit queries to the knowledge base.

Figure 2: Nave of Notre Dame de Paris, showing the repetition of elements. (Photograph by S. Van Liefferinge)

### 3.1 Superuser mode

A small set of superusers will be able to create and edit the generic model of a Gothic cathedral. This will consist of information about features generally considered typical of Gothic (such as the cruciform ground plan and use of pointed arches) as well as more common-sense information (such as the fact that the ceiling is above the floor). These are facts that are unlikely to be explicitly stated in an architectural description because the reader is assumed to know them already. Individual descriptions need only describe how a particular building differs from this generic model. The generic model will be underdetermined, in that it will remain silent about features that vary considerably across buildings (such as the number of vaulting units in the nave).

The generic description will be written in a domain-specific architectural description language (ADL) modeled on English, and translated into a logical programming language such as Prolog. The

A column is a type of support. Every column has a base, a shaft, and a capital. Most columns have a plinth. The base is above the plinth, the shaft is above the base, and the capital is above the shaft. Some columns have a necking. The necking is between the shaft and the capital.

Figure 3: Sample ADL listing.

general task of rendering the semantics of natural language into logic programming is addressed extensively by Blackburn and Bos (2005), and an architecture-specific treatment is given by Mitchell (1990). However, our goal is not a complete implementation of English semantics. Rather, our task is more like natural language programming, in which the computer is able to extract its instructions from human language. (For treatments of natural language programming systems in other domains, see Nelson (2006) and Lieberman and Liu (2005).) In particular, historical details, asides, and other language not pertaining to architecture would be treated as comments and safely ignored. A syntactic parser can extract those sentences and phrases of interest to the system and pass over the rest. The ADL should allow anyone reasonably familiar with architectural terminology to work on the description without the steep learning curve of a programming language. It should be able to understand multiple wordings for the same instruction, perhaps even learning new ones over time. As our eventual goal is to be able to understand real-world architectural texts, grammatical English sentences should not produce errors. Any such misunderstanding should be seen as an opportunity to improve the system rather than a failure on the part of the user. As an example of how a portion of a column description in an ADL might look, see Figure 3. In order to implement this ADL, a number of interesting problems must be solved. The following section describes a few we have dealt with so far.

**Referring to unnamed entities**

The simple statement "Every column has a base" does not have a straightforward rendering in a logical language like Prolog. In order to render it, we must be able to say that for each column, there exists some (unnamed) base belonging to that column. To do this, we use *Skolemization* (after Skolem (1928)), a technique for replacing existential quantifiers with unique identifiers (Skolem functions). Blackburn and Bos (2005) demonstrate the use of Skolem functions in capturing natural language semantics, and a contemporary application is demonstrated by Cua et al. (2010). Our implementation is a modified version of that described by Covington et al. (1988).

To say "Every column has a base", we insert two rules into the knowledge base. The first declares the existence of a base for each column:

```
base(base_inst(X, 1)) :- column(X).
```

The second tells us that the base belongs to the column:

```
has(X, base_inst(X, 1)) :- column(X).
```

Here `base_inst(X, 1)` is a Skolem function for an instance of `base`, where `X` is the name of the object to which it belongs, and `1` is its index. (In the case of a base, there is only one per column.) Thus a column named `column1` would have a base named `base_inst(column1, 1)`, and so forth.

**Context sensitivity**

Sentences are not isolated semantic units, but must be understood in terms of information provided by previous sentences. In the listing in Figure 3, the statement "the base is above the plinth" is interpreted to mean "each column's base is above that column's plinth". In order to make the correct interpretation, the system must know that the present topic is columns, and recognize that "base" and "plinth" are among the listed components of columns.

We assume the superuser's description constitutes a single discourse, divided into topics by paragraph. Accessibility domains correspond to paragraphs. When the description mentions "the base", it is assumed to refer to the base mentioned earlier in the paragraph as a component of the column. That the column is the paragraph's topic is indicated in the first sentence. Our treatment of discourse referents and accessibility domains is similar to that of discourse representation theory (Kamp and Reyle, 1993).

**Default reasoning**

We must have a way to dismiss facts from the knowledge base on the basis of new evidence. Our model describes the "typical" Gothic cathedral, not *every* Gothic cathedral. There is usually an exception to an apparent rule. To handle this, we make use of defeasible or nonmonotonic reasoning, as described by Reiter (1987) and Antoniou (1997). (Several variants of defeasible reasoning are also described by Billington et al. (2010).)

The ADL accommodates exceptions through the use of modifiers. Words like "all" and "every" indicate a rule that holds without exception. Words like "most" or "usually" indicate that a rule is present by default in the model, but can be altered or removed by future assertions. Finally, the word "some" indicates that a rule is not present by default, but can be added. The system's internal logical representation can keep track of which rules are defeasible and which are not. Attempts to make an assertion that conflicts with a non-defeasible rule will fail, whereas assertions contradicting a defeasible rule will modify the knowledge base. Conclusions derivable from the defeated rule will no longer be derivable. Our implementation is a somewhat simplified version of the system presented by Nute (2003).

**Partial ordering**

Defeasible reasoning can help us resolve a particular type of ambiguity found in natural language. Architectural descriptions contain many partial ordering relations, such as "above" or "behind". These relations are irreflexive, antisymmetric, and transitive. When such relations are described in natural language, as in the description in Figure 3, they are typically underspecified. We say that an item is "above" another, without making explicit whether it is immediately above. We also do not specify which is the first (e.g. lowest) element in the series. In our generic model, if it is simply stated that one item is above another, we insert a non-defeasible rule in the knowledge base, such as

above(capital, shaft)

The further assertion

immediately(above(capital, shaft))

is also made, but is defeasible. Should another item be introduced that is above the shaft but below the capital, the immediately relation no longer holds. We can also deal with underspecificity by recognizing when more than one state of affairs might correspond to the description. For example, if it has been asserted that item A is above item C, and that item B is above item C, we have no way of knowing the positions of A and B relative to each other. A query Is A above B? must then return the result maybe.

### 3.2 Administrator mode

The administrator mode is used to input information about particular buildings, as opposed to Gothic cathedrals in general. When an administrator begins an interactive session, the generic model designed by the superuser is first read into the knowledge base. The administrator simply describes how the particular cathedral in question differs from the generic model, using the same architectural description language. We would also like for the administrator mode to accept real-world cathedral descriptions in natural language rather than ADL. This is a nontrivial task, and complete understanding is likely a long way away. In the short term, the system should be able to scan a description, identify certain salient bits of information, and allow the administrator to fill in the gaps as needed. To illustrate the problem of understanding real-world descriptions, we present the following excerpt from a description of the Church of Saint-Maclou:

> The nave arcade piers, chapel opening piers, transept crossing piers, and choir hemicycle piers are all composed of combinations of five sizes of individual plinths, bases, and moldings that rise from complex socles designed around polygons defined by concave scoops and flat faces. All the piers, attached and freestanding on the north side of the church, are complemented by an identical pier on the opposite side. However, no two piers on the same side of the church are identical. (Neagley, 1998) p. 29.

There are important similarities between this description and our own architectural description language. We see many key entities identified (*nave arcade piers, chapel opening piers, etc.*), as well as

words indicating relationships between them (*composed, identical,* etc.) Even if complete understanding is not currently feasible, we could still use techniques such as named entity extraction to add details to our model.

### 3.3 User mode

The user mode will consist of a simple query answering system. Users will input queries such as "How many vaulting units are in the nave at Saint-Denis?" or "Show me all cathedrals with a four-story elevation." The system will respond with the most specific answer possible, but no more, so that yes/no questions might be answered with "maybe," and quantitative questions with "between four and six", depending on the current state of the knowledge base. Unlike web search engines, which only attempt to match particular character strings, our system will have the advantage of understanding. Since descriptions are stored as a logical knowledge base rather than a string of words, we can ensure that more relevant answers are given.

## 4 Conclusion

The ARC project is a great undertaking, and presents us with a number of problems that do not have ready solutions. We have presented just a few of these problems, and the techniques we have developed for solving them. There is still much work to be done in implementing the architectural description language, and processing real-world descriptions. In addition, there are some capabilities we would like to add to the system, such as producing graphical renderings from descriptions.

It is our hope that the ARC system, when completed, will be of great benefit to architectural historians, or anyone interested in Gothic cathedrals. Having a knowledge base of cathedral designs that can respond to queries will make the historian's task easier. The system's ability to identify vague or contradictory statements allows us to see how historical descriptions differ from one another. And the process of rendering architectural descriptions in a logical form could provide new insights into the design and structure of cathedrals.

## References


Grigoris Antoniou. 1997. *Nonmonotonic Reasoning.* The MIT Press, Cambridge, MA.

David Billington, Grigoris Antoniou, Guido Governatori and Michael Maher. 2010. An Inclusion Theorem for Defeasible Logics. *ACM Transactions on Computational Logic* Vol. 12, No.1, Article 6, October 2010.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics.* CSLI Publications, Stanford, California.

Michael A. Covington, Donald Nute, Nora Schmitz and David Goodman. 1988. From English to Prolog via Discourse Representation Theory. ACMC Research Report 01-0024, The University of Georgia. URL (viewed May 5, 2011): `http://www.ai.uga.edu/ftplib/ai-reports/ai010024.pdf`

Jeffrey Cua, Ruli Manurung, Ethel Ong and Adam Pease. 2010. Representing Story Plans in SUMO. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity.* Association for Computational Linguistics, Los Angeles, California, June 2010, 40-48.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Henry Lieberman and Hugo Liu. 2005. Feasibility Studies for Programming in Natural Language. *End-User Development. H. Lieberman, F. Paterno, V. Wulf, eds.* Kluwer, Dordrecht.

Yong Liu, Yunliang Jiang and Lican Huang. 2010. Modeling Complex Architectures Based on Granular Computing on Ontology. *IEEE Transactions on Fuzzy Systems,* vol. 18, no. 3, 585-598.

William J. Mitchell. 1990. *The Logic of Architecture: Design, Computation, and Cognition.* The MIT Press, Cambridge, MA.

Linda Elaine Neagley. 1998. *Disciplined Exuberance: The Parish Church of Saint-Maclou and Late Gothic Architecture in Rouen.* The Pennsylvania State University Press, University Park, PA.

Graham Nelson. 2006. Natural Language, Semantic Analysis and Interactive Fiction. URL (viewed May 5, 2011): `http://www.inform-fiction.org/I7Dowloads/Documents/WhitePaper.pdf`

Donald Nute. 2003. Defeasible Logic. In *Proceedings of the Applications of Prolog 14th International Conference on Web Knowledge Management And Decision Support* (INAP'01), Oskar Bartenstein, Ulrich Geske, Markus Hannebauer, and Osamu Yoshie (Eds.). Springer-Verlag, Berlin, Heidelberg, 151-169.

Raymond Reiter. 1987. Nonmonotonic Reasoning. *Ann. Rev. Comput. Sci.* 1987.2: 147-86.

Thoralf Skolem. 1928. Über die mathematische Logik (Nach einem Vortrag gehalten im Norwegischen Mathematischen Verein am 22. Oktober 1928). In *Selected Works in Logic.* Jens Erik Fenstad, ed. Universitetsforlaget, Oslo - Bergen - Tromsö, 1970, 189-206.

Eugène-Emmanuel Viollet-le-Duc. 1854-68. *Dictionnaire raisonné de l'architecture française du XIe au XVIe siècle.* vol. 2. Libraries-Imprimeries Réunies, Paris. Image URL (viewed May 5, 2011): `http://fr.wikisource.org/wiki/Fichier:Plan.cathedrale.Chartres.png`