Computational Richness in the Representation of Architectural Languages

Bige Tunçer, Rudi Stouffs

An extensive analysis of an architectural object or body leads to a model composed of abstractions, each reflecting on a different aspect. Though separately described through drawings, diagrams, and texts, these abstractions relate in many ways, most commonly through shared components. An integrated model that recognizes these relationships yields more than only the original abstractions. We present a methodology for achieving such a rich representation and touch upon the tools and techniques necessary to implement this methodology. As an application of this methodology, we describe an interactive educational system for the presentation of architectural analyses.

Keywords: architectural languages, abstractions, representational flexibility, meta-language, presentation

Introduction

In education, as in architectural history, theory, and design, complete and thorough analyses of architectural bodies and objects are indispensable. These analyses cover many different aspects of the subject, e.g., physical and contextual attributes as well as geometric, functional, typological and organizational relations. When the results are computationally integrated, new views and arguments can be deduced from these that transcend the individual abstractions. Achieving such an integration computationally is hampered by the fact that different informational aspects of an architectural system require different representational languages. We propose an alternative approach to the quest for a truly universal, computational language or model. This approach relies on a computational framework where each model or abstraction is specified in its own appropriate representation, though with a common syntax. Representations can be combined and integrated, and relationships between the abstractions expressed within this framework.

This computational framework defines a common syntax for the definition of representational languages and includes mechanisms for specifying and relating abstractions. Each abstraction is expressed as a hierarchical composition of structures at different levels and relationships between these structures. Each representation defines the same formal operations on its structures, allowing structures from different abstractions to be handled in a uniform way. Mechanisms for representing and relating abstractions rely on methodologies for recognizing patterns, variations, and regularities. These relationships collate the individual abstractions into an integrated model, from which new views and abstractions can be derived. In this way, a rich representation is achieved without imposing a fixed frame of reference.

The goal is to derive at a specific implementation, yet from general principles. The definition of abstractions and mechanisms can be interpreted and implemented for different architectural bodies. Therefore, the system must be flexible rather than
restrictive and while developed for a specific case study, i.e., Ottoman mosques, extendable to other examples.

**Background**

The Web offers many examples of architectural analyses on a wide variety of subjects. Commonly, these analyses consist of a collection of documents, categorized and hyperlinked to support navigation through the information space [i]. More sophisticated examples rely on a database for storage and management of the data, and offer a more complex categorization of the information entities and their relationships [ii].

These studies present effective ways of accessing and browsing information, however, within a fixed frame of reference. Their functionality can be improved by introducing a framework that allows abstractions to be represented each within their own frame of reference, and integrated at the same time.

**Architectural Languages**

The use of the concept of architectural languages seems indispensable in the conceptualization and description of buildings (Mitchell, 1990:100). An architectural language refers to formal and symbolic elements and to the relationships between them, and is characterized by a ‘vocabulary’ of elements and a ‘grammar’ whose rules indicate how these elements can be placed in space (Flemming, 1990:31).

The vocabulary elements are placed together with compositional principles or conventions that underlie a certain piece of architecture (or a collection of such pieces) and make it ‘recognizable’ (e.g. as belonging to a certain period, or as the work of a particular architect). It is commonplace, for example, to characterize architectural styles in terms of their vocabularies: the vocabulary of classical architecture includes the Doric column, the round arch, etc. In practice, a vocabulary does not need to be fixed, explicit, or well-defined but, in principle, the types comprising the vocabulary can be enumerated, named, and specified through type recognition mechanisms.

**Types**

In order to deal with meaning and form, methodologies in formalizing architectural languages should introduce ‘types’ into the description process. When type is considered as an artistic and intellectual manifestation of culture, a study of types, its internal rules, and the recognition of its instances becomes profitable in a comprehensive analysis process.

The typological studies of building morphology in historical analysis have established a rich body of architectural knowledge. According to Oxman (1990), the formal knowledge of architecture is defined in two parts: object descriptions achieved through formal analysis, and object-to-object relationships achieved through syntactic analysis. There are many examples of formal analysis in architecture in the literature. These analyses define common categories of architectural form, such as volumetric organization, circulation patterns, axes, and boundaries.

A type can be described as the encoding of the prominent features of a design object. Such features include the function, form, and context. A generalization and typification process makes it possible to formalize these features in a computational environment.

Moneo’s (1978) definition of type is the most relevant to the goal of this research. His definition is: “It [type] can most simply be defined as a concept which describes a group of objects characterized by some formal structure. It is fundamentally based on the possibility of grouping objects by certain inherent structural similarities. It might even be said that type is the act of thinking in groups.” According to Moneo, type as formal structure is “intimately connected with reality, with a vast hierarchy of concerns running from social activity to building construction. Ultimately, the group defining a type must be rooted in this reality as
well as in an abstract geometry. This means, for example, that buildings also have a precise position in history. ... This leads directly to the concept of a typological series that is generated by the relationship among the elements that define the whole. The type implies the presence of elements forming such a typological series and, of course, these elements can themselves be further examined and considered as single types; but their interaction defines a precise formal structure.”

The characteristics of types can be used to define the vocabulary by which the type can be described. As these characteristics are encoded in the type definition, later they can be used to deduce commonalities between different types. These commonalities can be used to define relationships between types, specifying the formal structure.

**Computational Richness in Representation**

**Concept**

Architectural objects are generally expressed through abstractions. A geometry, even augmented, specifies a single abstraction; other abstractions express other aspects of the object, such as function, acoustics, structure, process, form generation, space, and organizational relationships (Schmitt, 1993:39). Abstractions can be expressed as drawings, diagrams, models, images, simulations, and text.

An abstraction can be understood in a syntactic manner as a composition of structures and relationships between these structures. While each abstraction touches upon a different aspect, abstractions relate through commonalities, similarities, and variations in vocabulary, structures, and their relationships. When the abstractions are numerous and diverse, recognizing these relationships creates a tight network in which the individual abstractions no longer stand out. Such a network of abstractions can be said to embody a rich representation.

The representation of an abstraction requires the definition and recognition of the composing structures and relationships. Structures can be recognized as instances of types specified by the user or suggested by the system. Structures may be grouped into more complex structures, creating type (structure) to type (meta-structure) relationships. Structures may also belong to more than one meta-structure, in reference to the formal structure described by Moneo.

A syntactical framework that offers representational flexibility is needed to define the vocabularies that express these structures. Different abstractions require different vocabularies that have their origin in the domains of the respective abstractions. These vocabularies may overlap but, more often, they will offer alternative descriptions of related types reflecting on the function and context.

Search and recognition mechanisms can assist the user in relating structures within and between abstractions. Different mechanisms may be appropriate for different types of relationships. The following list of types is not exhaustive:

- Type relationships relate structures that belong to the same or a similar type. Relationships between types, e.g., a type is a subtype of another type, induce similar relationships between structures.
- Compositional relationships relate structures based on location. An example is the 'part of' relationship.
- Spatial relationships relate structures based on form.
- Referential relationships are common in text and hypertext, but can also be found in graphical information in the form of symbols.

The recognition of relationships between structures from different abstractions is complicated by the differences in representational languages and vocabularies. While the collation of vocabularies assists in the recognition of relationships between structures, relating types in different abstractions may also provide insights in the process of collating these
vocabularies. Mechanisms for relating structures may rely on simultaneous approaches on both levels.

The highlighted components in figure 1 are instances of three types: dome, courtyard, and fountain. All instances of the same type in the three abstractions (plan, section, and picture) are related through type relationships. Additionally, the courtyard has a compositional relationship to the fountain. This relationship is defined through the location of the fountain with respect to the courtyard, i.e., the fountain is in the center of the courtyard and the ratio of their sizes do not exceed a certain value.

Recognizing these relationships creates a complex network of structures and relationships that specifies an integrated model of the architectural object. The representation of this model specifies a
language and vocabulary for this model. From a representational point of view, this meta-language is a composition of the languages of the original abstractions. New abstractions can be considered as defined by new languages that form part of the meta-language. Such abstractions are important to the user in order to comprehend and interpret the resulting model. Rather than reducing the rich model into simpler abstractions, this ability to define new abstractions must be understood as queries to the model that are unrestricted by the original composition into abstractions.

Slicing the model for a new abstraction relies on the specification of a corresponding vocabulary. According to this vocabulary, types and their instances will be included or excluded from the section, resulting in a subset of the structures and relationships. These define the new abstraction.

Requirements

A computational framework that supports this methodology needs to fulfill some requirements about the representation of the abstractions, the recognition of structures and relationships, and the formal structure of the resulting model.

Instances of types recognized in the abstractions must be expressible in the representational vocabulary and recognizable within the structural forms of the representation. Since these structures may be hierarchically organized, where a structure is composed of sub-structures, the representation must allow for the recursive definition of structures. An automated recognition of these structures also depends on the initial representation of the abstractions. Often, instances of types may only be implicitly defined in the representation of the abstraction. Requirements on this initial representation depend on the recognition mechanisms adopted.

Different mechanisms and methodologies are suited to recognize structures within different abstractions, e.g., graphical information and text cannot be searched using the same methodologies. Some structures or abstractions are best handled by hand. In general, hybrid and semi-automatic methodologies will be necessary for optimal results. Within these methodologies, the user should be able to enumerate the types to be recognized, upon which the system stores these types for later use by the user.

The integrated model must be sufficiently rich, such that slicing the model yields sections whose structures are related in a non-trivial way. This means that both the number of abstractions must be sufficient and each abstraction must distinguish enough structures and relationships. Besides these recognized relationships, we consider additional, representational relationships that are implicit in the structural forms and relate structures that share representational components. It is unlikely that these representational relationships are made explicit as this would unnecessarily complicate the resulting model. A high number of representational relationships ensures that the model of structures and relationships does not degenerate into an object-oriented network that is more appropriately represented in a hypertext or hypermedia system.

Framework

We intend to explore the appropriateness of both sorts (Stouffs and Krishnamurti, 1997) and typed feature structures (Carpenter, 1992) as theories for the representational framework.

Sorts constitutes a representational formalism that specifies formal operations on sorts and recognizes formal relationships between sorts. Elementary data types are considered to define primitive sorts, which combine to composite sorts under formal operations defined over sorts. Important in the definition of a sort is a specification of the operational behavior of collections of elements of a sort, denoted forms, for common arithmetic operations. This behavioral specification enables a uniform handling of forms of different sorts.

The concept of sorts only provides for a common syntax, allowing for different vocabularies and languages to be created, compared, and related.
There is no imposition of concepts beyond the purely syntactical, and the alphabet of building blocks for the vocabulary definition can be readily extended at all times.

*Feature structures* are graph structures for presenting partial information that can be expressed in terms of features or attribute-value pairs. The value of a feature or attribute in a feature structure is either undefined or another feature structure. In *typed feature structures*, each feature structure is assigned a type, with types organized into a multiple inheritance hierarchy based on type inclusion. Feature structures are suited for use in unification-based formalisms, where consistent structures of partial information are combined into a single structure. Typed feature structures can only be unified if their types are compatible according to the primitive hierarchy of types.

Abstractions can be considered as partial information structures, the collating of abstractions as the unification of their corresponding typed feature structures. To implement such a system, one has to define the features and a corresponding type inheritance hierarchy.

**Tools**

The system must support the user to define, relate, and collate abstractions. Procedures to relate abstractions can rely on methodologies for recognizing patterns, variations, and regularities. Existing techniques, e.g. neural networks, can be adopted for recognizing parts of abstractions as instances of types.

Generally, four features are associated with neural networks (Tsoukalas et al., 1997:211): they learn by example; they constitute a distributed, associative memory; they are fault-tolerant; and they are capable of pattern recognition. Pattern recognition requires the neural network to match large amounts of input information simultaneously and generate a categorical or generalized input with a reasonable response to noisy or incomplete data. Neural networks are good pattern recognizers, even when the information comprising the patterns is noisy, sparse, or incomplete (Tsoukalas et al., 1997:213). Where there is poor or ambiguous input the system will always produce some sort of output from its memory. When the resulting match is not exact, we can reinforce the closest match by feeding the output pattern back into the system. Liu (1993) and Coyne (1990) present examples of studies in the field of architecture using neural networks for pattern matching.

Neural networks are also used in texts for recognition purposes (Greenberg, 1999). Neural networks and pattern recognition algorithms can pinpoint keywords in and extract key concepts from documents. This is done by determining which sets of text are related by identifying content patterns in one set and recognizing the same or similar patterns in other sets.

**Application**

A promising application of the methodology presented in this paper is an interactive educational system for the presentation of architectural analyses.

Architecture students often prepare case studies for their design studio projects, gathering information about existing buildings with similar functionality to the subject of their project. They present this in the form of collages on paper, or as hyper-documents. There have been attempts at collecting and organizing these results into computational environments. The EDAT example (Akin et al., 1997) additionally offers the students a tool to present their work in the design studio and is extendable in different ways, e.g., for carrying out performance analyses on the stored test cases. The methodology here presented can be understood as a further step from such approaches.

The realization of such a system leads to the following concerns:

- The amount of time needed to input the abstractions should be reasonable.
- The system should be easy to understand and interact with.
- The user interface should be easy to use.
We envision the system to be implemented as an application with a Web interface.

The input to such a system is a number of abstractions defined using a common syntax. From the usability point of view, there are two main actors: the creator and the user. The creator is the person who inputs the abstractions into the system, relates them, and collates them into an integrated model. The user is the person who accesses this model and derives new and original abstractions from the system.

Creator’s point of view: The creator first defines the vocabulary and specifies the types necessary for the representation of each abstraction. Using a variety of mechanisms, the system recognizes structures as instances of types and relationships between these, with assistance from the creator. Structures from different abstractions are related in the same manner. Once all the abstractions are represented and collated in the system, the model is ready for access.

User’s point of view: The user retrieves the individual (original) abstractions from the system and browses the information space. She is also able to select from the vocabulary offered by the system and derive a new abstraction specified by this vocabulary. According to this vocabulary, some types will be included and others not. The system should offer the user the possibility to further exclude certain types or abstractions from the result. This implies that the retrieval of a new abstraction is not a purely syntactic process, but also has a semantic component.

**Conclusion**

When thinking of representation and richness of representation with respect to architectural languages, instead of trying to achieve a single but complex model that encapsulates all the (computable) information, a better approach is to consider different models or abstractions of the same object. To support such an approach requires a representational framework that offers such flexibility and tools for representing and recognizing instances of types and relationships. The result is a rich model of the object that offers users the ability to query different aspects, unrestrained by the original abstractions.

**References**


Greenberg, Facing up to new interfaces, Computer 32(4) (1999) 14-16.


Acknowledgments

This paper is derived from the initial phase of the first author’s Ph.D. work at the chair for Architecture and CAAD, Swiss Federal Institute of Technology Zurich, under the supervision of Gerhard Schmitt, committee chair, Ramesh Krishnamurti (Carnegie Mellon University), and Sevil Sariyildiz (Delft Technical University).

Web References