CONCEPTUAL REQUIREMENT VALIDATION FOR ARCHITECTURE
DESIGN SYSTEMS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Gregory Flanagan

September 2011

© 2011

Gregory Flanagan

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Conceptual Requirement Validation for
Architecture Design Systems

AUTHOR: Gregory Flanagan

DATE SUBMITTED: September 2011

COMMITTEE CHAIR: Franz Kurfess, Ph.D.

COMMITTEE MEMBER: Mehul Bhatt, Ph.D.

COMMITTEE MEMBER: John Seng, Ph.D.

COMMITTEE MEMBER: Hisham Assal, Ph. D.

Abstract

Conceptual Requirement Validation for Architecture Design Systems

Gregory Flanagan

Computer-aided architectural design (CAAD) programs represent architectural design at a low level of spatial abstraction. While this representation model allows CAAD programs to capture the precise spatial characteristics of a design, it means that CAAD programs lack the underlying computational apparatus necessary to reason about design at a conceptual level.

This thesis is a first step towards building a framework that bridges the gap between the conceptual aspects of a design and its low level CAAD-based spatial representation. Specifically, this thesis presents a new framework, referred to as the Conceptual Requirements Reasoner (CRR), which provides an architect with a framework to validate conceptual design requirements. The CRR will demonstrate how qualitative spatial representation and reasoning techniques can be used as the link between a design's conceptual requirements and its underlying quantitative spatial representation.

A museum case study is presented to demonstrate the application of the CRR in a real world design context. It introduces a set of museum design requirements identified in research and shows how these requirements can be validated using the CRR. The results of the case study shows that the CRR is an effective tool for conceptual requirements reasoning.

Acknowledgements

Dr. Franz Kurfess: for all the guidance during my time as a graduate student and for supporting me while completing this thesis.

Dr. Mehul Bhatt: for the opportunity to work in the Design Space group and for all the guidance and advice formulating my ideas for this thesis.

My Parents: for all the support they have given me, especially during the last year as I completed this thesis.

Justin Phillips: for being a good friend and helping me finish this thing.

Dr. Carl Schultz: for encouragement and insight into the process of completing a thesis.

Contents

List of Tables	ix
List of Figures	x
1 Introduction	1
2 Background	4
2.1 Qualitative Spatial Representation and Reasoning	4
2.1.1 Qualitative Spatial Calculi	5
2.2 Constraint Logic Programming	7
2.3 Building Information Models	8
2.3.1 Generative Design	9
2.3.2 Pattern Languages	10
3 Related Works	11
3.1 Intelligent Critic System	11
3.2 The Architect’s Collaborator	12
3.3 SEED	13
3.4 Interior Design Configuration for a Virtual Environment	14
3.5 Others	15
3.6 Discussion	17
4 The Conceptual Requirements Reasoner	20
4.1 Design Problem	20
4.2 Scope and Contributions	24
4.3 Design and Approach	25

5	Design Representation and Spatial Reasoning	30
5.1	Design Representation	31
5.1.1	Physical Design Representation	31
5.1.2	Artefactual Design Representation	32
5.1.3	Implementation	34
5.2	Spatial Reasoning	37
5.2.1	Spatial Abstraction	39
5.2.2	Topology	41
5.2.3	Orientation	43
5.2.4	Distance	48
6	Design Space	50
6.1	Qualitative Spatial Attributes found in Architecture	51
6.1.1	Positioning	51
6.1.2	Positioning for Sequences	55
6.1.3	Facing	57
6.1.4	Proximity	59
6.1.5	Visibility	61
6.2	Architectural Concepts	64
6.2.1	Privacy	64
6.2.2	Continuity	66
6.2.3	Enclosure	67
6.2.4	Artefactual Requirements	69
6.3	Conceptual Requirement Validation	71
7	Museum Case Study	73
7.1	Museums	74
7.1.1	Museum Design Requirements	74
7.1.2	Maximizing Visitor Utilization of Exhibitions	75
7.1.3	Visitor Orientation in the lobby	80
7.2	Application of CRR for Museum Design	81
7.2.1	Museum Designs and Import	81

7.2.2	Validating Museum Requirements	84
7.2.3	Discussion of Museum Requirements	87
8	Conclusions & Future Work	90
8.1	Future Work	91
	Bibliography	94
A	Appendix	99
A.1	Spatial Abstraction of Architectural Entities	99

List of Tables

3.1	Related Work System Objectives	17
3.2	Related Work Features	18
3.3	Related Work Spatial Reasoning	19
7.1	Case Study Results	85
A.1	Spatial Abstractions	99

List of Figures

2.1	Qualitative Spatial Calculi	6
4.1	Architectural Levels of Abstraction - Source [7]	22
4.2	Design Space Multi-Hierarchy	26
4.3	Design Space Multi-Hierarchy	27
4.4	CRR Implementation Design	28
5.1	Artefactual Extension, Source [7]	32
5.2	Operational Space Interference	33
5.3	Representation Queries	35
5.4	Quality & Quantity Space	38
5.5	Topological Relationships	41
5.6	OPRA ₂ Space Partitioning	44
5.7	SCC Partitioning	45
5.8	Minimum Distance	48
6.1	Positioning	52
6.2	SCC Partitioning	53
6.3	Bedroom Positioning	55
6.4	facing	57
6.5	OPRA Partitioning	58
6.6	Door, Window Facing	59
6.7	Proximity	60
6.8	Desk Outlet Proximity	61

6.9	Visibility	62
6.10	Visibility Example: Doorway and Stairs	63
6.11	Privacy	65
6.12	Continuity	67
6.13	Enclosure, Confined and Open	69
6.14	Sink Doorway Interference	70
7.1	Results from Melton’s experiments on the effects of door position- ing in an Art Museum	76
7.2	Positioning of Doorways in Galleries	77
7.3	Display case arrangements	77
7.4	Display Case Sequence	78
7.5	Philadelphia Museum of Art	82
7.6	Lobby of Museu Calouste Gulbenkian	83
7.7	Asian Wing of Philadelphia Museum of Art	83
7.8	Visibility in Gallery239	86
7.9	Gallery Requirement Violations	87
A.1	Doorway Point Abstraction	100
A.2	Doorway Point Abstraction	101

Chapter 1

Introduction

Humans reason about space at an abstract level; this means we think about our spatial environments in terms of qualities rather than quantities. In other words, we cognitively represent our spatial environments using symbolic terms like *next-to*, *in-front-of*, or *close-by*, as opposed to using a precise spatial coordinate system of x, y, and z [20] [14]. Computers on the other hand are excellent at storing and computing over large amounts of precise, concrete data. From a spatial perspective, this means that computers are able to handle a large amount of quantitative spatial information [32] [30].

The area of architectural design is an interesting application domain with regards to the representation of space because it inherently involves multiple spatial perspectives: *qualitative* and *quantitative*. At the qualitative level, spatial aspects of the design define architectural qualities such as spaciousness, privacy, and continuity. These conceptual aspects encompass the way an architectural design is experienced. At the quantitative level, an architectural design is spatially modeled at a precise and concrete level so that it can be used during the construction of the design in the real world. These two spatial perspectives encompass very

different aspects of an architectural design, henceforth referred to as design.

Current state-of-the-art computer-aided architectural design (CAAD) programs spatially represent a design at a quantitative level [26]. This allows CAAD programs to be used effectively to model the precise spatial characteristics that are necessary during the construction of the design in the real world. Unfortunately, this type of representation model means that CAAD programs lack the underlying computational apparatus necessary to reason about design at higher levels of abstraction, such as reasoning about its conceptual features. This happens because of the representational gap between the low level spatial representation of the design in the CAAD program and the abstract representation necessary for reasoning at a conceptual level.

This thesis is a first step towards building a framework that bridges this gap between the high level conceptual aspects of a design to its low level CAAD-based spatial representation. Specifically, this thesis presents a new framework, referred to as the Conceptual Requirements Reasoner (CRR), that provides an architect with a framework to validate conceptual design requirements. Conceptual design requirements in this thesis refer to qualitative and experiential features of a design. They can be architectural concepts like spaciousness, privacy and continuity or qualitative spatial attributes found in architecture (QSA) such as positioning, orientation, proximity, and visibility. The CRR will demonstrate how qualitative spatial representation and reasoning techniques can be used as the link between a design's conceptual requirements and its underlying quantitative spatial representation.

A Museum Case Study is presented to demonstrate the application of the CRR in a real world design context. It introduces a set of museum design requirements that were identified in museum research and applies these requirements to two

real world museums. The results of the case study show that the CRR is an effective tool for conceptual requirements reasoning.

This thesis is organized as follows. Chapter 2 presents the background knowledge that is necessary for the rest of the thesis. Chapter 3 presents the related works. Chapter 4 introduces the problem statement being addressed in the thesis and presents the its main contributions and approach. Chapters 5 presents the CRR's design representation model and spatial reasoning framework. Chapter 6 presents a set of architectural concepts and qualitative spatial attributes found in architecture and shows how they can be spatially defined. Chapter 7 presents the application of the CRR in the area of museum design through a case study and reports the results of the study. Chapter 8 presents concluding thought and future works to be completed.

Chapter 2

Background

This chapter presents the background information required for this thesis. There are three areas of background knowledge that will be required for the this thesis: Qualitative Spatial Reasoning (QSR), Constraint Logic Programming (CLP) and Building Information Modeling (BIM).

2.1 Qualitative Spatial Representation and Reasoning

The field of Qualitative Spatial Reasoning (QSR) investigates techniques for representing and reasoning about abstract features of space in a computational framework while preserving its fundamental properties. In this regard, QSR represents space at a qualitative level as opposed to a quantitative level. At the qualitative level space is defined by symbolic relationships, while at the quantitative level space is defined by precise numeric values [20] [7].

Qualitative representations of space can be quite useful. Take for example the

task of describing the location of a chair to your friend. In order to communicate the location of the chair effectively it is unnecessary to use precise numeric values that characterize the chair, such as its size, shape and weight, and metric spatial measurements such as distance that relate the chair to other objects. Communicating qualitative spatial relationships, such as the orientation of the chair, is sufficient for effectively describing its location. An example could be, “the chair is *behind* the table and on the *left-hand* side of the room.” By using qualitative relationships, you are representing space at a level of abstraction that hides unnecessary details so that you are dealing only with those aspects of space that are important for the problem at hand.

While the example above is informal, formal qualitative representations of space preserve universal spatial aspects such as uniqueness (objects exist exactly once), and realizability while providing a computational apparatus to reason about space from an abstract level [20]. The following subsection will introduce several formal qualitative spatial representation calculi with respect to topology and orientation.

2.1.1 Qualitative Spatial Calculi

Formal qualitative spatial calculi are explicit characterizations of qualitative spatial relationships from a given perspective, eg. topology and orientation, which logically define intrinsic spatial properties such as spatial uniqueness and spatial composition. Examples of qualitative spatial calculi that are important for this thesis include the Region Connection Calculus (RCC) [36] for topological relationships, the Single Cross Calculus (SCC) for extrinsic orientational relationships [21] and Oriented Point Relation Algebra (OPRA) [33] for intrinsic orientational

Figure 2.1: Qualitative Spatial Calculi

relationships.

RCC defines eight topological relationships over regions: disconnected, partial overlap, externally connected, equals, non-tangential proper part, inverse non-tangential proper part, tangential proper part, and inverse tangential proper part. Refer to Figure 2.1a for an illustration of these relationships. Axiomatic rules of jointly exhaustive, mutually disjoint and pairwise disjoint (JEPD) enforce the spatial uniqueness of each relationship [36]. For example, two regions can *not* be partially overlapping and disconnected at the same point in time. Additionally, composition tables define the set of realizable spatial configurations for a collection of regions. For example, if region A is inside region B and region B is inside region C , then region A must also be in region C .

Oriental relationships can be broken down into intrinsic and extrinsic spatial relationships. Extrinsic relationships are defined by an external reference point. This type of orientational relationship is characterized in the SCC, in which an external reference point is used to relate two objects together. Figure 2.1b illustrates the SCC partitioning of space for three points, A , B , and C . In this example, C is related to A through B , and has an SCC 2 relationship.

Intrinsic relationships are defined by an internal perspective. This type of relationship is characterized in OPRA, in which the relationship of an external

object is defined by the internal direction of the observing object. Figure 2.1c shows the relationship between two points, A and B . In this example point B has an OPRA_2 relationship with A of 7 and point A has an OPRA_2 relationship with B of 1 .

As with RCC, axiomatic rules of JEPD and composition tables enforce spatial properties of uniqueness and composition for these two orientational calculi.

2.2 Constraint Logic Programming

Constraint Logic Programming (CLP) combines logic programming with constraint solving [25] [42] [27]. Logic programming is based on a declarative semantic in which programs are defined in terms of *what* should be computed instead of *how* it should be computed (i.e. procedural style). In this regard, a logic programming language relies on built-in search and unification algorithms to solve logic based formula. At its essence, logic programming can be viewed as a basic form of constraint solving [25].

Prolog is one of the most well known and used logic programming languages. It is based on predicate logic, unification, logic conjunction, and depth-first search [39]. From a constraint perspective, Prolog solves constraints over the set of uninterpreted structures, i.e. Prolog's unification algorithm works entirely on a syntactic level. What this means is that Prolog does not internally represent data beyond its syntax and can only unify terms based on this. Take for example the arithmetic expression $5 + 1 = 6$, where in Prolog this expression will fail because equality is true only on the syntactical level, i.e. $5 + 1 = 5 + 1$.

Constraint solving techniques can help logic programming by increasing the

expressiveness of the language beyond the syntactic. In general, constraint solving is based on constraints, which are simply restrictions over the values a set of variables can bind to. For specific constraint solvers, the variables and the constraints over the variables are based on a specific domain, such as arithmetic expressions, boolean logic, etc., which encode the semantics and unification algorithms for solving the logical formula directly in a logic based language. In the example above, a constraint solver for arithmetic expression would be able to resolve the expression $5 + 1 = 6$ as true because it knows directly about the semantics of addition and equality.

Building new constraint solvers is laborious because it typically requires hardwiring the constraint solver into the compiler or interpreter [25]. The use of Constraint Handling Rules (CHR) can help this process by providing a high level programming framework designed for the implementation and integrate constraint solvers into logic based languages. The aim is to ease the process of building constraint solvers in terms of defining a new domain of constraints and integrating these constraints into a logic programming framework [24] [23]. CHR works by using logic based rules that rewrite constraints via simplification and propagation rules and constraints are solved by continually rewriting constraints until they are in a solvable form. Because these rules are based on logic themselves it integrates seamlessly into other logic based frameworks.

2.3 Building Information Models

Building Information Modeling (BIM) brings together multiple semantic perspectives of the building process and life-cycle into a single informational data source [18] [17]. Compared to CAAD tools which represent building elements as

generic geometric primitives devoid of semantic context, BIM provides a mechanism to represent complex semantic properties and relationships between building elements. The goal of which is to provide a breadth of building information which can be used in building analysis (lighting, fire, energy consumption, construction costs, etc.), and used throughout the entire building life cycle. Through the use of BIM, architects are able to build and use tools which allow them to represent and reason about aspects of the building that go beyond pure geometric constraints.

One of the main objectives of BIM is to promote interoperability between building tools. To this end the Industry Foundation Classes (IFC) were developed as an open data modeling specification for BIM [22]. Within the context of this thesis the IFC data format is used to interface between CAAD and the Conceptual Requirements Reasoner.

2.3.1 Generative Design

Generative approaches to design explore the possibilities of a design as defined by a set of constraints [18]. To do this, generative design parametrically tags building objects (walls, doors, windows, etc.) with behavioral and/or functional properties and constraints. Based on these properties and constraints new designs and/or design changes are automatically generated. For example, when the context of a building element changes, such as its load balancing properties, material composition, or spatial relationships to other building objects, new design layouts are generated in order to preserve parametric constraints.

2.3.2 Pattern Languages

A pattern language for architecture defines a syntax and grammar that encompass architectural design patterns and rules. For architecture, Alexander [3] defined a set of over 200 patterns which inform architects and designer to architectural design best practices. These patterns distill abstract architectural concepts into concrete, pragmatic rules which can be followed to create an environmental property. For example, Alexander defines a pattern for privacy and how it corresponds to intimacy. The pattern states that:

Unless the spaces in a building are arranged in a sequence which corresponds to their degree of privateness, the visits made by strangers, friends, guests, clients, family, will always be a little awkward.

Alexander goes on further to explain that privateness can be achieved by the placement of rooms within a house, such that the porch, common spaces, and kitchen are centrally located near the entrance of the house, while private spaces such as the bedroom and office are located on the periphery of the house, removed from the main entrance. The crux of the work here demonstrates that abstract architectural concepts can be directly linked to concrete properties of the building and defined as a set of design rules.

Chapter 3

Related Works

This chapter presents the related work for this thesis. There is a large body of research on tools for conceptual design assistance. The related works that will be discussed in this chapter have been narrowed to tools that provide assistance to a designer and use spatial reasoning as the basis of its design analysis.

3.1 Intelligent Critic System

The intelligent computer-aided architectural design system (ICAAD) [13] is a tool that evaluates architectural floor plans based on government regulations and interior design guidelines. The research here focuses on the spatial representation model for architectural designs in the context of checking for requirements. In this respect, ICAAD represents designs in an object oriented hierarchical model, which is grounded to architectural containment relationships, i.g., windows are contained by rooms, rooms are contained by apartments, and apartments are contained by buildings. Various architectural entities, such as rooms, windows and furniture are all represented individually as ICAAD objects in the hierarchical

structure. Each ICAAD object is responsible for storing spatial data about the entity, including dimension, shape, and direction. Spatial relationships between entities, such as proximity and visibility, are extracted from the spatial data and are represented as a network, which connects entities together based on their spatial relationships.

ICAAD uses Prolog’s inference engine to match design rules (i.e., government regulations and interior design guidelines) against the ICAAD representation model. Violations are detected when a spatial relationship between entities is matched to a design requirement rule in Prolog.

ICAAD’s focus and main advantage is on the representation of design requirements and their definition in terms of qualitative spatial relationships. To this end ICAAD provides a specification for an architectural design hierarchy and mechanisms to detect design violations. However, ICAAD has several limitations which make it difficult to be used in a real world architectural work flow. To begin with, qualitative spatial relationships between architectural objects need to be manually generated. This means that there is no link between the quantitative model expressed in a CAAD tool and ICAAD. Additionally, ICAAD uses ad hoc definitions for qualitative spatial relationships which makes it difficult to evaluate the validity of the relationships.

3.2 The Architect’s Collaborator

The Architect’s Collaborator (TAC) [29] is a design support system that evaluates designs based on architectural qualities and provides recommendations to the designer with respect to these qualities. Within TAC, architectural qualities refer to experiential aspects of a design such as *openness* and *privacy*. The main

contribution of TAC is that it evaluates designs based on architectural qualities and provides recommendations for improving the design.

For example, a designer might use TAC to evaluate a kitchen based on it being *visibly open*. TAC evaluates this quality (visibly open) based on a pre-defined spatial characteristic of the design, in this case, the percentage of this room (kitchen) that is visible from an adjacent room. A room is considered by TAC to be visibly open if the result of the calculation is above a certain threshold. If it is below the threshold, TAC will provide recommendations to the designer to help increase the visual openness of the kitchen.

TAC's main advantage is that it focuses on a single architectural quality (openness) and provides a breadth of analysis and metrics for evaluating a design based on it. No other system provides the amount of analysis on a single quality that TAC does. However, The main drawback to TAC is its use of 'black box' spatial reasoning. From the example above, TAC directly maps the architectural quality of visual openness to a specific low level geometric characteristic of the design. This type of spatial reasoning is very specific to the definition of the problem, therefore making it difficult to expand.

3.3 SEED

The Software Environment for Supporting the Early Phases in Building Design (SEED) [2] project is a framework that supports an architect during the entire architectural design process. Specific to the work done in this thesis, the SEED project introduced the idea of using constraints to represent design knowledge in order to generate floor layouts automatically.

Within the SEED project, design knowledge is partitioned into two categories: design units and functional units. Design units represent the spatial and physical elements of a building. Functional units represent design requirements that constrain design units based on spatial attributes such as shape, size and placement, and non-spatial attributes such as material make-up.

Given a design, the SEED project automatically generates possible floor plans that adhere to all of the design requirements, given a set of design and functional units. The main contribution of this work is that the system utilizes the constraint solving abilities in a CLP framework to autonomously find design layouts that meet the design’s requirement specifications. However, SEEDS is limited to generating basic floor layouts that are based on quantitative spatial constraints. This means that requirements have to be defined quantitatively.

3.4 Interior Design Configuration for a Virtual Environment

Calderon *et. al* [11] proposed a system that automatically configures an indoor virtual environment with respect to the placement interior design elements, by specifying the design problem in terms of spatial constraints. The system works by solving constraint satisfaction problems as specified over the space of the virtual environment.

The spatial constraint problems are defined as follows: Given a space for the virtual environment represented as a discrete Cartesian grid, a set of interior design objects (desks, chairs, lamps, etc.), and a set of spatial constraints over the design objects (e.g., a desk should be at least four meters from a power

outlet), the system searches for viable locations for each object within the virtual environment such that all of the spatial constraints are satisfied.

The main contribution of this work lies in the use of spatial constraints for reasoning about virtual environment design. Specifically, this work leverages the constraint solving capabilities of a CLP framework to discover new design layouts. However, the work here is preliminary and has several shortcomings. To begin with, the virtual environment used in this system is limited to a rectangular grid represented by discrete Cartesian coordinates. This means all spatial reasoning is over the discrete finite integer domain. Additionally, the expressiveness of the spatial constraints is limited. For example, topological relationships are defined over rectangles and points with respect to the point being inside or outside the rectangle. No consideration is made for orientation, or objects with richer geometries than points and rectangles.

3.5 Others

The following research presents other related work that is similar to the work in this thesis, but differs from the primary works discussed above because they are less established works.

Bhatt *et al.* proposed a system in [6] that validates a designer’s conceptual requirements against an architectural design in the context of Smart Homes. This research introduced the idea of modeling the artefactual space of the design’s structural elements and uses topological reasoning to detect design inconsistencies per a set of design requirements.

Key proposed a descriptor language [28] that defines architectural qualities

from a computational perspective. The work here is quite similar to the work in this thesis with regards to the approach that architectural qualities emerge from spatial configurations of design entities and that these qualities can be spatially characterized. The language defines architectural descriptors for concepts such as enclosure, continuity, and spaciousness and does so by interpreting the descriptor in terms of spatial properties. However, the approach here lacks the integration of a qualitative intermediate layer between the high level concepts and low level geometries and instead uses a more direct approach to geometric and spatial reasoning.

Borrmann introduced the idea of including predefined spatial constraints directly into a standard building production model, the Industry Foundation Classes (IFC) [22], to detect constraint violations [10]. This proposal focuses primarily on the definition and integration of spatial constraints into the IFC data model and also provides some discussion on how these spatial constraints might be checked in such an environment. While similar to CRR, this proposal's approach is focused on the integration of spatial constraints into a specific building model (IFC) and does not go into the process of generating the spatial constraints from the spatial configurations of a design.

The Intelligent Computer-Aided Design System (ICADS) [34] is a collaborative design support tool that assists a designer by providing functional based analysis of a design. The tool provides analysis to a designer based on several different aspects, such as lighting, noise insulation, climate control, energy conservation, space layout, and construction costs. The focus of this research is on developing a series of knowledge based software agents that collaborate within ICADS to make recommendations to the user.

3.6 Discussion

Table 3.1 shows the high level objectives of each tool discussed in this section. This table includes requirements checking, design evaluation, and automatic floor plan configuration. Despite the differences, each system provides design assistance at a conceptual level.

	CRR	ICAAD	TAC	Bhatt	ICAD	SEED	CLP
requirement checking	X	X		X			
design evaluation			X		X		
automatic floor plan configuration						X	X

Table 3.1: Related Work System Objectives

Out of this group the ICAAD tool has the most in common with the CRR. Both systems provide a framework for checking design requirements using qualitative spatial relationships. Despite these similarities, there are many differences. ICAAD focuses on the spatial representation model for designs in the context of requirement reasoning and gives little attention to or discussion of the underlying techniques for spatial reasoning and spatial abstraction, as is the focus in the CRR. While the ICAAD approach allows the system to reason automatically over the entire design, it does so by relying on the underlying 'black box' spatial reasoning. Extending 'black box' reasoning can be difficult because spatial functions are built only to solve a specific problem. On the other hand, the CRR uses general qualitative spatial reasoning and formal spatial calculi to reason over architectural designs. The advantages to this approach is that the reasoner is extensible, i.e., it is easy to build new design requirements by using the existing services of the spatial reasoners.

Table 3.2 shows various features provided by the related works. This includes

	CRR	ICAAD	TAC	Bhatt	ICAD	SEED	CLP
experiential qualities	X	X	X				
artifactual extension	X			X			
spatial representation and reasoning	X	X	X	X	X	X	X
formal qualitative spatial calculi	X			X			
feedback / recommendation			X		X		
design evaluation			X		X		
constraint based reasoning	X					X	X

Table 3.2: Related Work Features

whether the tool supports reasoning about experiential qualities (openness, privacy), and artefactual extension (operational / functional space), if the tool incorporates spatial representation and reasoning techniques and formal qualitative spatial calculi, if the system provides the user with feedback or recommendations based on design analysis, and if the system uses constraint based reasoning.

The CRR provides provides five of the seven features. It does not provide feedback to the user and it does not evaluate a design based on metrics. The intention for this version of CRR is to provide a solid framework for spatial reasoning over architectural designs and requirement reasoning. The omission of these features was based on time constraints and not any fundamental issues with the framework.

Table 3.3 shows the spatial reasoning capabilities for each system. It includes spatial reasoning for topological relationships, orientation (intrinsic and extrinsic), distance, and metric. Metric relationships refer to quantitative geometric functions such as the calculation of length or area.

	CRR	ICAAD	TAC	Bhatt	ICAD	SEED	CLP
topology	X		X	X		X	X
intrinsic orientation	X	X					
extrinsic orientation	X						
distance	X	X	X			X	X
metric	X	X	X		X	X	X

Table 3.3: Related Work Spatial Reasoning

The CRR is the only tool in the group that provides support for every type of spatial reasoning. Additionally, it is the only system that does so using formal qualitative spatial reasoning and representation techniques. This is one of the main advantages of the CRR. Using a broad variety of spatial relationships (topological, orientational, distance, metric) the CRR provides flexibility and a rich set of spatial relationships to build design requirements. This is important because the spatial configurations that make up a building are complex and need flexibility in terms of the way they are characterized and reasoned over.

Chapter 4

The Conceptual Requirements Reasoner

This chapter is an introduction to the Conceptual Requirements Reasoner (CRR), a framework for validating conceptual design requirements against CAAD-based design plans. The chapter begins by presenting the problem statement that is addressed in this thesis. The next section presents the scope and contribution of the thesis. The final section introduces the high level design and implementation of the CRR in the context of the problem statement.

4.1 Design Problem

The field of Spatial Computing for Architectural Design [7] investigates the computation apparatus that is necessary to represent and reason about architectural form and function from multiple levels of abstraction: *conceptual*, *qualitative*, and *quantitative*. At the conceptual level the architect perceives the design at a high level of abstraction. For example, at this level the architect would think

in terms like spaciousness, privacy and continuity. At the quantitative level the architect engages the design at a low level of abstraction. Now the architect considers concrete, physical aspects of the design. These include the geometric shape and spatial location of the design’s structural form. These levels of abstraction are referred to in this thesis as the architectural design hierarchy, where the top of the hierarchy corresponds to the conceptual (design), the middle corresponds to the qualitative, and the bottom corresponds to the quantitative.

Applying the architectural concept of spaciousness to the architectural design hierarchy will show how high level concepts are connected to low level spatial quantities. In this example, the concept of spaciousness conveys an experiential quality of the design such as the feeling of openness. Such an experiential quality emerges from the spatial design patterns of the building’s physical form. In other words, the location and placement of architectural entities such as walls, ceilings, windows, and doors, and their spatial relationships as they define the size and shape of rooms and spaces, directly affect the experiential quality of openness. In general, changes in the spatial patterns at the quantitative level affect the experiential qualities of the design at the conceptual level. Figure 4.1 illustrates this concept in which the features of the design at the conceptual level emerge from its spatial environment at the quantitative level via spatial patterns on the qualitative level.

Unfortunately, state-of-the-art CAAD programs [26] do not support the representation of and reasoning about architectural design at a conceptual level. This is because the CAAD-based spatial representation model is encapsulated at the quantitative level of the architectural design hierarchy, which lacks the semantics necessary for representation and reasoning at higher levels of abstraction. While this model is well suited for certain design tasks, such as the building of an

Figure 4.1: Architectural Levels of Abstraction - Source [7]

accurate blueprint of the design, it does not support other architectural design processes that require reasoning at a higher level of abstraction. These include design conceptualization, requirements reasoning and functional specification, to name a few [7].

Take for example the tasks an architect might face during the design conceptualization phase of a museum, including, among other things, defining high level functional and aesthetic requirements of the museum's design. The following descriptions of the *Museu Calouste Gulbenkian* in Lisbon, Portugal exemplifies this phase [41]:

The current topographic conditions of the site, where the larger trees are located in an area that is more elevated than the whole northern rim of the plot, made it possible to install a huge underground floor in the already existing depression, whose covering creates a gentle artificial elevation that perspectively accentuates and enhances the whole architectural composition. The distribution of the construction volumes fundamentally followed a desire for horizontality, allowing one to read the continuity of the green space beyond the construction and in all directions.

This sober, rational and markedly horizontal structure is distinctive for its laminar exterior, its modular repetition, austere design and the

hard quality of the materials which shape it, concrete and glass.

Environmental aspects of the museum’s design, such as these expressed in the above descriptions, encompass high level spatial design patterns that are identified at the design conceptualization phase. Because CAAD-based programs do not represent the design at a level of abstraction that is required to express these conceptualization, they are therefore incapable of providing the designer with assistance during this phase of the design process.

Design reasoning at a conceptual level, such as the example above, requires the underlying model representation semantics at the same level of abstraction. In this regard, the design semantics at the conceptual level involve abstractions over architectural entities such as rooms, floors, buildings, doors, and walls, along with the relationships they share between each other, both spatially (facing, near-to, across-from, connected-to) and non-spatially (material composition, lighting attributes, and load bearing qualities). Conversely, the underlying model representation semantics of CAAD-based program are based on spatially general and low level geometric primitives of points, lines, and polygons that lack the notions at the higher levels of the architectural design hierarchy. Whereas an architect might conceptualize the design of a museum in terms such as galleries and the relative placement of objects within each gallery, e.g., “the statue is on the *left-hand-side* of the gallery as one enters from the main doors,” the process of building the museum in a CAAD-based program restricts the architect to using points, lines and polygons.

The *semantic gap* exemplified between the conceptual and quantitative levels of the architectural design hierarchy is the crux of the problem being addressed in this thesis. The following section introduces the scope of this thesis by nar-

rowing the focus of the thesis to conceptual requirements reasoning, outlines the approaches taken towards this goal, and presents the thesis’s main contributions.

4.2 Scope and Contributions

The objective of this thesis is an initial step towards building a framework that bridges the gap between a design’s conceptual features to its real world quantitative representation (i.e., how it is represented in a CAAD-based program). Specifically, the framework, referred to as the Conceptual Requirements Reasoner or CRR, does this in the context of conceptual requirements validation. Conceptual requirements validation involves checking that the underlying spatial structures of a design, in the quantitative representation, are consistent with its conceptual features as they are specified by an architect. Towards this goal the following objectives have been identified as crucial:

- the identification, formal modeling, and spatial interpretation of architectural concepts
- the qualification of CAAD-based spatial models, and
- the use of qualitative spatial representation and reasoning techniques to link high level concepts to low level spatial quantities

To address these objectives, this thesis follows the approach introduced by Bhatt and Freksa [7], with regards to Spatial Computing for Architectural Design. This thesis does so by following the design space hierarchy as illustrated in Figure 4.1, in implementing the CRR. The main contributions in this respect are identified here:

- the implementation of a framework that validates conceptual design requirements against a CAAD-based spatial representation
- the identification of a set of QSA and architectural concepts
- the development of a multi-hierarchy that maps QSA and architectural concepts to spatial patterns in the qualitative level
- the implementation of spatial constraint solvers that qualify CAAD-based spatial models
- the development of a real world museum case study that demonstrates the applicability and effectiveness of the CRR in validating museum design requirements

The following section introduces the high level design and implementation of the CRR with regards to the scope and contributions identified in this section.

4.3 Design and Approach

The design and implementation of the CRR is based on the design hierarchy as illustrated in Figure 4.2. Following this structure, the CRR breaks down the problem into a design space module, quality space module, and quantitative space module. At the top level, the design space model is responsible for the representation of and reasoning about architectural concepts and QSA. This module will be used as the basis for building conceptual requirements. The middle level of the hierarchy, the quality space module, represents the design based on qualitative spatial relationships of orientation, topology and distance. Within the CRR,

Figure 4.2: Design Space Multi-Hierarchy

qualitative spatial relationships are defined using formal qualitative spatial calculi such as the Region Connection Calculus for topological relationships, the Single Cross Calculus for extrinsic orientational relationships and the Oriented Point Relational Algebra for intrinsic orientational relationships. The lowest level of the hierarchy, the quantity space module, is responsible for representing the design in terms of its physical and artefactual form.

Given these three modules, the main issue now becomes how each level interfaces with the others. In other words, how do concepts in the design space relate to qualitative spatial relationships in the quality space, and similarly, how do qualitative spatial relationships in the quality space relate to geometric objects in the quantity space? At the core of this is the qualitative module, an intermediary layer that interfaces between the design and quantitative modules.

Architectural concepts and QSA are linked to qualitative spatial relationships of orientation, topology and distance by explicit characterization. In other words,

Figure 4.3: Design Space Multi-Hierarchy

concepts in the design space are explicitly defined by qualitative spatial relationships. For example, the architectural spatial quality of positioning, where objects can be spatially situated at the same or opposing sides of a room, is defined by extrinsic orientational spatial patterns using the OPRA calculus. Chapter 6 will go into further detail regarding the specific spatial interpretation and characterization of each concept and QSA as identified in the thesis.

The geometric quantities in the quantitative level are connected to the qualitative spatial relationships in the qualitative level through spatial abstraction and qualification. Through this process the spatial quantities of the design are connected to qualitative spatial relationships. Chapter 5 will go into the details regarding the CRR's underlying spatial representation structure at the quantitative level and the spatial reasoning necessary to spatially abstract and qualify the quantitative representation.

The implementation of the CRR is composed of three modules as identified

Figure 4.4: CRR Implementation Design

in the design hierarchy, and is illustrated in Figure 4.4. The top level module is where architectural concepts and QSA are defined in terms of qualitative spatial patterns and is the interface through which users will query the CRR. The quantitative space module models the design’s physical form and artefactual extension. The data at this level is imported from a multi-modal access framework [37] based on the IFC standard for CAAD-based systems. The qualitative space module is responsible for the spatial abstraction and qualification of the quantitative model and is implemented as a series of spatial constraint solvers for topological, orientational, and distance reasoning.

From a user perspective, the CRR provides the user with design assistance in validating requirements for a work-in-progress design. The CRR does this by

providing a query engine that allows a designer to validate conceptual requirements. For example, a designer is provided with a set of architectural concepts and QSA, which the designer uses to construct queries to the CRR. So if a designer wants to check if a door and window are facing each other (as might be a requirement for air circulation purposes) the designer can make the following query to the CRR:

```
?- facing(door, window).
```

The CRR will respond *true* to the query if the door and window are facing each other and respond *fail* if they are not.

Chapter 5

Design Representation and Spatial Reasoning

This chapter presents the quantity and quality modules of the CRR. For the quantity module, this chapter introduces the internal architectural design representation model. For the quality module, this chapter introduces the spatial constraint solvers that are used by the CRR to qualify spatial relationships. Additionally, this chapter presents each module in the context of the declarative programming paradigm used for the implementation of the CRR, namely Prolog and Constraint Logic Programming (CLP).

This chapter has two sections. The first section presents the internal architectural design representation structure and shows how it is implemented in Prolog. The second section looks at the spatial constraint solvers and shows how each is implemented in CLP.

5.1 Design Representation

The CRR represents two distinct aspects of a design. It represents the design’s physical form and non-physical artefactual extensions. The physical form pertains to the precise geometric data that defines the design’s architectural entities with regard to layout, location and shape. In this thesis architectural entities refer to the structural elements of walls, doors, windows, slabs, stairs, and columns. Artefactual extension pertains to the non-physical but geometric data that defines the functional characteristics of architectural entities, such as the space used by a person to operate a doorway.

The physical representation of the design is automatically imported into the CRR from a multi-modal spatial data access framework [37] built by the DesignSpace project [16]. For each architectural entity, the CRR imports the structural geometric coordinate data, as it is represented in a CAAD-based program, and a unique identifier referred to in the CRR as the *err.id*. Currently, artefactual extensions are manually imported, but the intent for future versions is to automate this directly from the multi-modal spatial data access framework.

The remainder of this section is comprised of three subsections. The first subsection looks at the physical representation structure. The second subsection looks at the artefactual representation structure. The third subsection looks at how both are implemented in the CRR.

5.1.1 Physical Design Representation

The CRR represents a design’s physical form in terms of geometric features of layout, shape, and location. Each architectural entity is represented in the

CRR as a polygon composed of points that correspond to the physical dimensions (length, width, shape) of the entity as it occurs in the real world. The physical representation of the design is analogous to the design perspective that is represented by CAAD programs.

5.1.2 Artefactual Design Representation

An artefactual extension is a non-physical, spatial extension of an architectural entity that pertains to a specific functional quality. While an artefactual extension does not have a physical manifestation, it is nonetheless an important aspect of the design representation model because it affects the functionality and performance of a design from an experiential perspective.

There are three types of spatial artefacts identified by Bhatt[6] that are represented in the CRR: *operational space*, *functional space*, and *range space*.

Figure 5.1: Artefactual Extension, Source [7]

The operational space encompasses the region of space required by an object to perform its function. For example, the operational space of a door is the entire space which a door can occupy during its operation, opening and closing

(see Figure 5.1b). This space is dynamic in the sense that it represents the configurable space of the physical doorway.

The functional space encompasses the region of space that is required for an object to function itself or the space for a person to operate an object. For example, the functional space of a doorway is the total space required for a person to operate, open and close, as well as any additional space need to navigate entirely through the doorway (see Figure 5.1b).

The range space pertains to a region of space that encompasses the range of a sensor (i.e., camera, motion sensor). For example, the range space of a camera is defined by its depth of field and angle of view. It represents the total space that is under surveillance of the camera (see Figure 5.1b).

Figure 5.2: Operational Space Interference

Each artefactual extension is represented in the CRR as a polygon that corresponds to its spatial dimensions.

Artefactual extensions directly affect the experiential quality of a design. For example, a sink in a bathroom has a functional space, the space a person occupies while washing his/her hands at the sink. If this space interferes with (i.e., topologically overlaps) the functional and/or physical space of a doorway, this will cause a circulation problem when both the doorway and sink are simultaneously

being used, because it will cause a collision between the person washing his/her hands and the doorway. In this example, the functional aspects of the bathroom's design with regard to the mitigation of circulation problems is directly influenced by the spatial configuration of the non-physical artefactual extensions of the design. It should be noted that this type of occurrence can not be modeled in current CAAD-based programs because they do not represent artefactual extension.

5.1.3 Implementation

Within the CLP framework of the CRR, the physical and artefactual representations are implemented as factbases and rulebases. There are two factbases that are in-memory database-like structures. One of these factbases is used as a catalog, which relates each architectural entity to its architectural type (wall, window, door, room, interior design object, etc.) via its *crr_id*. Each entry in this factbase is a Prolog predicate of the form *architectural_type(crr_id, type)*.

```
architectural_type(door42, door).
architectural_type(wall49, wall).
```

The other factbase is used as a linking table to relate architectural entities together based on containment relationships. The CRR represents two types of containment relationships. The first is the containment relationship between rooms and architectural entities and interior design elements. This containment relationship defines which architectural entities and interior design elements are contained in which rooms. In a specific design, *desk2* is contained in *room6*, and *wall5* is contained by *room6*. Each containment relationship is implemented as a Prolog predicate of the form *contained_room(contained_crr_id, containing_crr_id)*.

```
contained_room(desk2, room6).
contained_room(wall5, room6).
```

The second containment relationship is between doors / windows and walls. This containment relationship defines which doors / windows are contained by which walls. For example, *door42* is contained in *wall6*. This is implemented as a Prolog predicate of the form *contained_wall(contained_crr_id, containing_crr_id)*.

```
contained_wall(door42, wall6).
```

Figure 5.3: Representation Queries

Rulebases link architectural entities to their physical and artefactual representations. There is a separate rulebase for the physical, functional, operational, and range representations that are implemented using a similar form. Each rulebase is built as a series of Prolog rules. The head of each rule links an architectural entity to a variable that is bound to its spatial representation. The body of each rule is responsible for handling coordinate point data by internally defining spatial primitives of points, lines, and polygons. The *spatial_primitive* rule transforms coordinate data into spatial primitives and binds a variable to a spatial representation. The *spatial_primitive* rule works by matching arguments. For example, the following call to the *spatial_primitive* rule with a single point in the second argument instantiates the variable, *SpatialPrimitive*, as a point object primitive.

```
spatial_primitive(SpatialPrimitive, (5,7)).
```

Similarly, a call to *spatial_primitive* with a list of points instantiates the variable as a polygon object primitive.

```
spatial_primitive(SpatialPrimitive, [(5,7), (5,9), (7,7)]).
```

The following rules show the physical, functional and operational rules for a door with a *curr_id* of *door42*. Note the similarities and differences between rules.

```
physical_representation(door42, Geometry) :-  
    spatial_primitive(Geometry, [(1,3), (1,4), (4,4), (4,3)]).
```

```
functional_representation(door42, FuncGeometry) :-  
    spatial_primitive(FuncGeometry, [(1,7), (4,7), (1,1), (4,1)]).
```

```
operational_representation(door42, OpGeometry) :-  
    spatial_primitive(OpGeometry, [(1,4), (4,6), (4,3)]).
```

Every architectural entity has a physical representation but may or may not have a functional, operational, or range representation. For example, a doorway usually has a functional and operational representation but may not have a range representation. Some entities, such as columns, have no artefactual representations. However, there are no limitations imposed on which entities can or can not have certain artefactual extensions.

These factbases and rulebases can be queried using a Prolog interpreter to retrieve information about architectural entities. A call to the *architectural_type* factbase can be used to get an entity's type given its *curr_id*, such as the following query.

```
architectural_type(door42, Type).
```

The return of this call instantiates the *Type* variable with the door's type, which is *door*. Similarly, a query to a rulebase will return an instantiated variable

with the respective spatial representations. Figure 5.3 shows the implementation structure for querying the rulebases. For example, the following query to the *physical_representation* rulebase will instantiate the *Geometry* variable with *door42*'s physical representation.

```
physical_representation(door42, Geometry).
```

Calls to the functional, operational, and range rulebases work in the exact same fashion.

5.2 Spatial Reasoning

Qualitative spatial representation and reasoning is used in the CRR as a link between a design's high level conceptual aspects and its quantitative spatial representations. This process begins by spatially abstracting the design's quantitative spatial representation into spatially abstract objects that are used by the spatial constraint solvers to qualify topological, orientational, and distance relationship over the design, as is illustrated in Figure 5.4.

Figure 5.4: Quality & Quantity Space

These spatial constraint solvers use formal qualitative spatial calculi for topological (RCC), orientational (SCC, OPRA) and distance to qualitatively represent and reason over a design. Within the CLP framework of the CRR, the spatial constraint solvers are implemented using $\text{CLP}(\mathbb{R})$ (CLP over Reals) and Constraint Handling Rules (CHR). The topological and distance constraint solvers extend Sindalog [4], a spatial constraint solver written in CHR.

Each spatial constraint solver is implemented as a query engine that qualifies spatial relationships to answer qualitative spatial queries. A qualitative spatial query involves three steps:

1. retrieval of quantitative representation
2. spatial abstraction of quantitative representation
3. qualification of spatial relationships

As an example, the following topological query involves the inside relationship between a desk and a room.

```
?- inside(desk3, room1).
```

In step one, the quantitative geometries of the desk and room are retrieved from the physical representation rulebase. The second step uses transformation rules (discussed in the next section) to spatially abstract the physical representation into abstract spatial objects. In the final step, the abstract objects are used by the topological constraint solver to qualify the inside relationship. The constraint solver will respond *true* if the desk has an inside spatial relationship with the room, and *fail* if it does not.

The remainder of this section is structured as follow. The first subsection looks at how the quantitative representations are spatially abstracted. The second, third and fourth subsections look at topological, orientational and distance based reasoning.

5.2.1 Spatial Abstraction

Spatial abstraction is the process of generalizing a spatial representation so as to represent only those features that are necessary for solving a specific problem. Within the context of the CRR, quantitative spatial representations can be abstracted to points, directed points and convex hulls in order to reason about the design's qualitative spatial relationships of topology, orientation and distance. For example, one approach to reasoning about the qualitative orientational relationship between two doors requires that both doors be spatially represented as directed points. Because doors are naturally represented as polygons, spatial abstraction is needed to transform their quantitative polygonal representation into a qualitative directed point representation.

The process of spatially abstracting an architectural entity depends on the

following:

1. the type of spatial transformation, eg. line to point, polygon to convex hull
2. the interpretation of the intrinsic nature of an entity within an architectural and design context

For example, the abstracted representation of a doorway as a directed point is not only defined by its physical form (a polygon) but also by the interpretation of the doorway’s intrinsic direction in the context of its situation in the design. Further discussion on this topic can be found in [A.1](#).

Within the CLP framework of the CRR, architectural entities are spatially abstracted using Prolog rules. These rules transform an entity’s quantitative spatial representation into abstracted objects of points, directed points and convex hulls. Transformation rules take a *crr_id*, retrieve the quantitative geometry and perform a series of geometric transformations based on the dependencies discussed earlier. For example, the rule to spatially abstract a door into a point involves first retrieving the door’s physical representation and then transforming the door’s polygonal representation into a point by calculating the polygon’s centroid.

```
point_abstraction(Id, PtAbs) :-  
    physical_representation(Id, Polygon),  
    centroid(Polygon, PtAbs).
```

Given a *crr_id*, a query to an abstraction rule will return an instantiated variable with the entity’s spatially abstracted geometry. For example, in the following query the point abstraction for *door55* is bound to the *PtAbs* variable.

```
point_abstraction(door55, PtAbs).
```

5.2.2 Topology

The CRR can qualify four topological relationships for convex hulls: inside (5.5a), outside (5.5b), partially overlapping (5.5c), and disconnected (5.5d).

An object A has an inside relationship with an object B if all A 's points are inside the convex hull of B . The inside relationship is qualified by checking the property of convex hull to point containment. A point in A is contained inside the convex hull of B if the polarity of the determinate between a point in A and each line segment in B is negative [15]. In the CRR, points in a convex hull are always ordered counterclockwise, therefore the polarity of the determinant between a point inside the convex hull will always be negative.

An object A has an outside relationship with an object B if all B 's points are inside the convex hull of A . Because the outside relationship is the inverse of the inside relation it is easy to qualify by using the inverse of the inside relationship ($inside(B, A)$).

An object A has a partially overlapping relationship with an object B if one or more line segments in A intersect with one or more line segments in B . The partially overlapping relationship is qualified by checking for line segment intersection between objects. Two objects are partially overlapping if they intersect in at least one point.

Figure 5.5: Topological Relationships

An object A has a disconnected relationship with an object B if A does not intersect with B , there are no points in A that are inside B , and there are no points in B that are inside A . The disconnected relationship is qualified by checking for line segment intersection and convex hull to point containment. If there are no line segment intersections and no convex hull to point containment relationships, then two objects are disconnected.

Spatial Constraint Solver: Topology

The topological constraint solver qualifies topological relationships over convex hull objects. Its implementation extends the Sindalog [4] spatial constraint solver by integrating topological constraints over convex hulls. It is written using CHR that simplify and propagate topological constraints into geometric constraints of line segment intersection and convex hull to point containment. For example, qualification for the partially overlapping topological relationship involves solving line segment intersection constraints between two convex hulls. The partially overlapping relationship is solved to true when there is at least one line intersection between the convex hulls and it fails when there are none.

The implementation of the partially overlapping relationship begins with a top level constraint that recursively propagates line segment intersection constraints between each line segment in one of the convex hulls. In the following constraint the first argument represents a line segment (two points) and the second is a convex hull (list of points)

```
intersect_segment_hull((Pt1, Pt2), [Pt3, Pt4|PS]) <=>
    intersection_segment_segment((Pt1, Pt2), (Pt3, Pt4)),
    intersection_segment_hull((Pt1, Pt2), [Pt4|PS]).
```

The *intersection_segment_segment* constraint represents the intersection between

two line segments and is in a solvable form. The *intersect_segment_hull* constraint is a recursive call that will propagate the *intersection_segment_segment* constraint for the rest of the convex hull. The qualification for the other topological relationships is similar to the partially overlapping constraint presented here, but in addition to the line intersection constraint some relationships use the geometric properties of convex hull to point containment.

The topological constraint solver is used in the CRR to query topological relationships over a design. Given two *crr_id*'s any of the four topological queries can be made:

```
?- inside(Id1, Id2).
?- outside(Id1, Id2).
?- partially_overlapping(Id1, Id2).
?- disconnected(Id1, Id2).
```

Because the topological constraint solver does not know anything about architectural entities there is an intermediate layer between a topological spatial query and the topological constraint solver that is responsible for retrieving quantitative geometries and transforming them into convex hull abstractions. The following rule handles queries for the inside topological relationship.

```
inside(Id1, Id2) :-
    convexhull_abstraction(Id1, Convex1),
    convexhull_abstraction(Id2, Convex2),
    insdie_convexhull(Convex1, Convex2).
```

5.2.3 Orientation

The CRR can qualify two types of orientational relationships, intrinsic and extrinsic. Intrinsic orientational reasoning is based on the $OPRA_m$ calculus. $OPRA_m$ partitions space into half-planes, which are defined by lines emanating

from a reference point, and defines the qualitative orientational relationships by the half plane in which a second point is located in relation to the reference point. The granularity of an OPRA_m relationship, specified by m , defines the number of lines emanating from the reference point.

Figure 5.6: OPRA_2 Space Partitioning

Figure 5.6 shows the OPRA_2 (two lines) partitioning of space, with labels for the base OPRA_2 relationships. Point A is located in half-plane 7 of point B 's partitioned space, and B is located in half-plane 1 of A 's partitioned space. Note that the arrow depicts the point's direction and the half-planes are labeled starting at the point's direction, incrementing counter-clockwise.

OPRA_m relationships are qualified in the CRR by solving a system of linear inequalities, which are based on the lines that emanate from the reference point. Given a point A and B , the θ_{AB} symbol refers to $\tan^{-1} \frac{B_y - A_y}{B_x - A_x}$, the ϕ_A symbol refers to the direction of point A , and the m symbol refers to the granularity of the OPRA_m relations. Equation 5.1 shows the system of linear inequalities that qualify OPRA_m relationships as referenced from point A to point B . Using this system of linear inequalities, the OPRA_m relationship (i) can be solved for B in

terms of A 's OPRA_m space.

$$\begin{aligned} & \left(\left(\left(\frac{i}{2} \in N \wedge i < 2 \right) \wedge \left(2\pi - \frac{i-2}{4m} < \theta_{AB} - \phi_A < \frac{2\pi}{4m} \right) \right) \right) \\ \vee & \left(\left(\left(\frac{i+1}{2} \in N \wedge i \geq 1 \right) \wedge \left(\theta_{AB} - \phi_A = 2\pi \frac{i-1}{4m} \right) \right) \right) \end{aligned} \quad (5.1)$$

Extrinsic orientational reasoning is based on the SCC. The SCC defines the orientation of a point C (referent) with respect to a point B (relatum) from the viewpoint of a point A (origin).

Figure 5.7: SCC Partitioning

Figure 5.7 shows an SCC relationship between three points. The partitioning scheme is based on the orientation of the origin and relatum points. In this example, the third point, the referent, is located in half-plane 5 of the SCC

partitioned space, so it has an SCC_5 relationship with point A as related by B .

$$\begin{aligned}
& \left(\left(\left(A_y < B_y \right) \wedge \left(C_y > M_{\text{perp}} \cdot C_x + Y_{\text{perp}} \right) \right) \vee \right. \\
& \left. \left(\left(A_y > B_y \right) \wedge \left(C_y < M_{\text{perp}} \cdot C_x + B_{\text{perp}} \right) \right) \right) \\
& \wedge \left(\left(\left(A_x < B_x \right) \wedge \left(C_y < M_{AB} \cdot C_x + B_{AB} \right) \right) \vee \right. \\
& \left. \left(\left(A_x > B_x \right) \wedge \left(C_y > M_{AB} \cdot C_x + B_{AB} \right) \right) \right)
\end{aligned} \tag{5.2}$$

SCC relationships are qualified by solving a system of linear inequalities, which are based on the lines that partition space into the SCC half-planes. Labeling starts at the line that runs between the origin and relatum, starting at zero and incrementing clockwise. Given three points A , B , and C , M_{AB} refers to the slope of line from A to B , Y_{AB} refers to the y-intercept of the line from A to B , M_{perp} refers to the slope of the line that is perpendicular to the line from A to B , and Y_{perp} refers to the y-intercept of the line that is perpendicular to the line from A to B . Given a point C , it has an SCC_1 relationship with points A and B if the equations in 5.2 can be solved. If the equations are not solvable then C does not have an SCC_1 relationship with A related to B . There are similar systems of inequalities for each of the seven SCC relationships.

Spatial Constraint Solver: Orientation

The orientational constraint solver qualifies OPRA_m and SCC relationships over points and directed points. It is written in $\text{CLP}(\mathbb{R})$ and uses constraints over linear inequalities (Equations 5.1, 5.2) to solve orientational qualification problems. Within the orientational solver there are two modules, one for intrinsic relationships based on OPRA_m and one for extrinsic relationships based on SCC.

The OPRA_m constraint solver solves extrinsic orientation qualification prob-

lems by constraining the linear inequalities in Equation 5.1. Given an origin point A , a reference point B , and a granularity m , the constraint solver will return the OPRA_m relationship (i) between A and B . The linear inequalities are constrained as $\text{CLP}(\mathbb{R})$ constraints and return once all variables in the linear inequalities are instantiated.

The SCC constraint solver solves intrinsic orientation qualification problems by constraining the linear inequalities in Equation 5.2. Given an origin, a relatum and a referent, the SCC constraint solver will return true or fail based on the resolvability of the linear inequalities. The linear inequalities are constrained as $\text{CLP}(\mathbb{R})$ constraints and will return true if the linear equalities are solvable and fail if they are not.

The orientational constraint solver is used in the CRR to query orientational relationships over a design. Given two or three *crr_id*'s the following orientational queries can be made:

```
?- scc(crr_id1, crr_id2, crr_id3, base_relation).
?- opra(crr_id1, crr_id2, m, i).
```

Because the orientational constraint solver does not know anything about architectural entities there is an intermediate layer between an orientational spatial query and the orientational constraint solver that is responsible for retrieving quantitative geometries and transforming them into point and directed point abstractions. The following rule handles queries for the scc and opra relationships.

```
scc(Id1, Id2, Id3, BR) :-
    point_abstraction(Id1, Pt1),
    point_abstraction(Id2, Pt2),
    point_abstraction(Id3, Pt3),
    scc(Pt1, Pt2, Pt3, BR).

opra(Id1, Id2, m, i) :-
```

```

directed_point_abstraction(Id1, DPt1),
directed_point_abstraction(Id2, DPt2),
opra(DPt1, DPt2, m, i).

```

5.2.4 Distance

The CRR can calculate the minimum euclidean distance between points, lines, and polygons. Given two objects the minimum distance is calculated by 1) finding the closest point in each object to the other object and 2) calculating the minimum euclidean distance between the two points.

Figure 5.8: Minimum Distance

Spatial Constraint Solver: Distance

The distance constraint solver calculates minimum distance queries and uses the distance constraint services of the Sindalog [4] spatial constraint solver for all distance calculations. The distance constraint solver is different from the other two because it returns a specific value, instead of returning *true* or *fail* based on a query. For example, the following query will return the minimum distance between the two objects in D.

```
min_distance(Id1, Id2, D).
```

The distance constraint solver is written using CHR that break down distance constraints until they can be solved for. This process involves two steps. The

first step calculates the point for each object that is closest to the other object,
and second step calculates the minimum distance between these two points.

Chapter 6

Design Space

This chapter presents the design space module of the CRR. It identifies a set of qualitative spatial attributes found in architecture (QSA) and a set of architectural concepts and show how both can be spatially interpreted using formal qualitative spatial calculi. Additionally, it demonstrates how they can be used to validate conceptual design requirements.

QSA and architectural concepts were ascertained from literature on architectural design [40] [3] [5] and museum research on visitor behavior [31] [9] [19] [38]. Their definitions are explicit characterizations of spatial design concepts that are based on heuristic architectural knowledge and common sense notions of spatial relationships. The objective here is not to present an exhaustive discussion of spatial design concepts or a validation for their spatial interpretation; rather it is to demonstrate how each can be represented and reasoned about in the CRR.

This chapter is structured as follows. The first section presents a set of QSA and defines each using formal qualitative spatial calculi. The second section presents a set of architectural concepts that are defined using QSA and spatial

relationships. The third section demonstrates how conceptual design requirements can be validated within the framework presented in this chapter.

6.1 Qualitative Spatial Attributes found in Architecture

This section presents a set of QSA and formally defines each QSA using qualitative spatial calculi. QSA are qualities that emerge from a design's spatial characteristics. In general they are defined by the spatial relationships between architectural entities. They can be a perceptual quality that is described from a specific vantage point, or an intrinsic quality that is inherent in the design's spatial structure.

This section presents five QSA: positioning, positioning for sequences, facing, visibility, and proximity. The goal of this section is to develop a set of QSA that qualitatively describe architectural designs from an experiential perspective.

6.1.1 Positioning

The positioning attribute defines an orientational relationship between a pair of objects as each object relates to the other within an extrinsic context, i.e., a defined space such as a room, apartment or building. There are four positioning relationships using this scheme: opposing-side of space, same-side of space, left-side of space, and right-side of space. The opposing-side and same-side relationships can be combined, via conjunction, with the left-side or right-side relationships to form the following compound relationships: opposing-left-side, opposing-right-side, same-left-side, same-right-side. Figure [6.1](#) shows an illustra-

Figure 6.1: Positioning

tion of the positioning attribute.

The positioning attribute is interpreted spatially using the SCC. As discussed earlier, the SCC defines the orientation of a point C (referent) with respect to a point B (relatum) from the viewpoint of a point A (origin). The positioning attribute uses the SCC partitioning scheme to orientationally relate two points (origin, referent) in the context of an external frame of reference (relatum). In the CRR, the reference point is defined as the centroid of the context space. By partitioning space using the centroid, an approximation is made that divides the space into the positioning relationships. An assumption made here is that this space will generally be rectangular in shape.

Figure 6.2: SCC Partitioning

Using this scheme, two points are defined as being on opposing sides of a space if they have an SCC relationship of 0, 1, 2, 6 or 7, as related by the centroid of the space. Conversely, a pair of points are defined as being on the same side of a space if they have an SCC relationship of 3, 4, or 5, as related by the centroid of the space. The SCC relationships of 2 and 6 could have been defined for either positioning attribute; the convention here is that they represent an opposing-side relationships. A point is defined as being on the left side of a space from another point if it has an SCC relationship of 5, 6, or 7, and is on the right side if it has an SCC relationship of 0, 1, 2, 3, or 4. Again, the SCC relationships of 0 and 4 could have defined either left or right; the convention here is that they represent the right-side attribute. Using the union set operation these relationships can be combined to form the opposing-left/right, and same-left/right relationships.

These positioning relationships are implemented in the CRR as Prolog rules. There is a single rule for each relationship: opposing-side, same-side, left-side, and right-side. The head of each rule takes three arguments: the first two are `crr_ids` for the architectural entities being related and one for the context space. The body of the rule defines the SCC relationships of each positioning quality.

The following is Prolog code for opposing-side and same-side positioning rules.

```
opposing_side(Id1, Id2, Id3) :-
    ( scc(Id1, Id2, Id3, scc0);
      scc(Id1, Id2, Id3, scc1);
      scc(Id1, Id2, Id3, scc2);
      scc(Id1, Id2, Id3, scc6);
      scc(Id1, Id2, Id3, scc7) ).

same_side(Id1, Id2, Id3) :-
    ( scc(Id1, Id2, Id3, scc3);
      scc(Id1, Id2, Id3, scc4);
      scc(Id1, Id2, Id3, scc5) ).

left_side(Id1, Id2, Id3) :-
    ( scc(Id1, Id2, Id3, scc5);
      scc(Id1, Id2, Id3, scc6);
      scc(Id1, Id2, Id3, scc7) ).

right_side(Id1, Id2, Id3) :-
    ( scc(Id1, Id2, Id3, scc0);
      scc(Id1, Id2, Id3, scc1);
      scc(Id1, Id2, Id3, scc2);
      scc(Id1, Id2, Id3, scc3);
      scc(Id1, Id2, Id3, scc4); ).
```

Conjunction can be used to build the same-left, same-right, opposing-left and opposing-right rules.

```
opposing_left(Id1, Id2, Id3) :-
    opposing_side(Id1, Id2, Id3),
    left_side(Id1, Id2, Id3).

same_right(Id1, Id2, Id3) :-
    same_side(Id1, Id2, Id3),
    right_side(Id1, Id2, Id3).
```

The positioning attribute can be used to query if architectural entities or interior design elements are on the opposing or same, left, or right side of a space to the other. For example, a design requirement might state that a bedroom

Figure 6.3: Bedroom Positioning

should be positioned to the back of a house from the main entrance. The reason for this requirement is that rooms that are in the back of a house are more private than rooms in the front of the house. In this requirement, the origin is the entrance, the referent is the bedroom, and the relatum is the centroid of the house. Figure 6.3 shows a bedroom that is on the opposing side of an apartment from the main entrance.

6.1.2 Positioning for Sequences

The positioning for sequences attribute defines a perceptual orientational relationship for a sequence of objects as each object relates to a vantage point in an extrinsic context. A sequence is a set of architectural entities, or interior design objects that are grouped together. In general, sequences are grouped together based on the proximity (near to each other) but any relationship can define a sequence. There are two positioning relationships for a sequence of objects: horizontal perceived and vertical perceived.

The positioning for sequence of objects attribute can be spatially interpreted using the SCC. This quality uses the SCC partitioning scheme to orientationally

relate a sequence of points (origin, referents) in the context of an external frame of reference (relatum). In the CRR, the reference point is defined as the centroid of the context space. By partitioning space using the centroid, an approximation is made that divides the space into the positioning relationships. As with the positioning attribute, an assumption made here is that the space will generally be rectangular in shape.

Using the SCC partitioning of space, a sequence of objects is perceived horizontally from the vantage point if at least one point in the sequence is on the left-side of the space and at least one point is on the right. A sequence of points is perceived vertically if they have the following combination of SCC relationships: at least one point in the sequence is on the opposing side of the room and at least one point in the sequence is on the same side of the room. Sequences can be both vertical and horizontal or they can be neither.

The positioning for a sequence of objects relationships are represented in the CRR as Prolog rules. There is a single rule for each relationship: horizontally perceived and vertically perceived. The head of each rule takes three arguments, the first for the the vantage point, the second is the context space and the third represents the sequence of objects (list of crr_ids). The following is Prolog code for the horizontal positioning rules. The rule uses two recursive helper predicates (left_r and right_r) to iterate through the sequence of objects.

```
horizontally_perceived(Id1, Id2, Seq) :-
    left_r(Id1, Id2, Seq),
    riht_r(Id1, Id2, Seq).

left_r(_, _, []) :- fail, !.

left_r(Id1, Id2, [Id3, Rest]):-
    ( left_side(Id1, Id2, Id3) -> true ;
      left_r(Id1, Id2, Rest) ).
```

```

right_r(_, _, []) :- fail, !.

right_r(Id1, Id2, [Id3, Rest]) :-
    ( right_side(Id1, Id2, Id3) -> true ;
      right_r(Id1, Id2, Rest) ).

```

6.1.3 Facing

The facing attribute defines an intrinsic orientational relationship between two objects as each object relates to the other. The definition used here is based on a common sense interpretation of facing, in which an object A is facing towards an object B , if A is directionally oriented at B . Under this definition, a pair of objects can be facing towards or away from each other. Given a pair of objects, A and B , there are four possible facing configurations between the pair:

Figure 6.4: facing

The facing attribute is spatially defined in the CRR using the Oriented Point Relation Algebra (OPRA). The advantage of using OPRA is that it models the intrinsic orientational relationships between a pair of directed points. Using OPRA_8 , the facing attribute can be defined in terms of the half plane sectors. Point A is facing towards point B , if B is located in sectors 0, 1, or 31 of A 's OPRA_8 partitioned space. Similarly, if A is located in sectors 0, 1, or 31 of B 's OPRA_8 partitioned space then B is facing towards A . In this case, both points

Figure 6.5: OPRA Partitioning

are facing towards each other. Conversely, point A is facing away from point B if B is located in sectors 2 - 30 of A 's partitioned space. A granularity of 8 is used in OPRA (i.e., OPRA_8) to define a narrow interpretation of the facing attribute.

Figure 6.5a shows an example of two points facing towards each other; note that each point is located in sector 31 of the other's partitioned space. Figure 6.5b shows an example of two points that are facing away from each other and are not located in sectors 0, 1, 31 of the other's partitioned space.

The facing attribute is implemented in the CRR as Prolog rules. There is one rule for facing-towards and one for facing-away. The head of each rule takes two arguments for the architectural entities being related. The body of the rule is a single disjunctive clause that checks for the OPRA_8 relationship between the architectural entities. Given two architectural objects, the rule returns true if the architectural entities are facing towards or away from each other as defined by the rule, and fails if they are not. The following is Prolog code for the facing rules.

```
facing_towards(Id1, Id2) :-
    ( opra(Id1, Id2, 8, 0) ;
      opra(Id1, Id2, 8, 1) ;
      opra(Id1, Id2, 8, 31) ).
```

```
facing_away(Id1, Id2) :-
```

```
( facing_towards(Id1, Id2) -> fail; true).
```

In the CRR, the facing attribute can be used to check if two architectural entities or interior design elements are facing towards or away from each other. For example, doorways and windows that face towards each other promote airflow through the space they inhabit. Figure 6.6 shows two example designs of a room with a single window and doorway. Figure 6.6a shows the configuration where the doorway and window are facing towards each other, while Figure 6.6b shows the configuration where the doorway and window are facing away from each other.

Figure 6.6: Door, Window Facing

6.1.4 Proximity

The proximity attribute defines a qualitative distance relationship between two objects. In the CRR, there are three proximity relationships: near, near+, and far. The spatial interpretation of proximity is defined in the context of architecture, which is restricted here to be *within a building*. Based on this interpretation, proximity is formally defined using heuristics based on what is considered near, near+, and far within a building.

Figure 6.7: Proximity

Using these heuristics, two objects are near if the minimum distance between them is less then or equal to 6 feet, near+ if the minimum distance between them is between 6 - 12 feet, and far if the minimum distance between them is greater than 12 feet.

Proximity is represented in the CRR as three Prolog rules. There is one rule for near, one for near+, and one for far. The head of each rule takes two arguments for the architectural entities being related. The following is Prolog code for the proximity rules.

```
near(Id1, Id2) :-
    minimum_distance_compare(Id1, Id2, <=, 6).

near_plus(Id1, Id2) :-
    minimum_distance_compare(Id1, Id2, <=, 12),
    minimum_distance_compare(Id1, Id2, >, 6).

far(Id1, Id2) :-
    minimum_distance_compare(Id1, Id2, >, 12).
```

The proximity attribute can be used to check whether two architectural entities are near or far. For example, a computer desk should be located near a

power outlet to make it easy to power up the desk's computer and desk's lamp. Figure 6.8 shows a desk that is located near a power outlet and another that is not.

Figure 6.8: Desk Outlet Proximity

6.1.5 Visibility

The visibility attribute defines a boolean relationship between a viewpoint and an object with respect to the object being visible from the viewpoint. This interpretation of visibility is based on the topological characteristics of an object blocking (obstructing) a view. It does not take into account the material characteristics of the obstructing object, such as glass, which would allow visibility through the obstructing object. Additionally it does not take into account the effects of distance on visibility, i.e., an object that is very far away is not visible. However, within the context of architecture, the magnitude of distances is limited to a point that it is not a factor in this interpretation of visibility. Under this definition, an object is visible if the view-space, the space between the viewpoint and the object, is not obstructed.

The spatial interpretation of visibility is based on the topological character-

istics of the space in-between the viewpoint and object. This space is referred to as the view-space. An object is visible if the view-space is not obstructed by surrounding structures or objects. An obstructing object is an object that has a partially overlapping or containing topological relationship with the view-space. Artefactual extension is not considered as an obstructing object because it does not have a physical form. This definition does not distinguish between visibility that is partially or totally obstructed.

Figure 6.9: Visibility

Visibility is represented in the CRR as a single Prolog rule. The head of the rule takes two arguments, the viewpoint and the object that is being checked for visibility. The body of the clause calculates the view-space and checks the surrounding objects for a topological obstruction. Surrounding objects are architectural entities that are contained in the same room as the vantage point and object. If the vantage point and object are in different rooms the object is not visible.

Given a viewpoint and an object, the rule returns true if the object is visible,

and fails if it is not. The following is the Prolog code for the visibility rule.

```
is_visible(Id1, Id2) :-
    entities_in_same_room(Id1, Id2, Entities),
    generate_viewspace(Id1, Id2, ViewSpace),
    obstructed_r(ViewSpace, Entities).

obstructed_r(_, []).

obstructed_r(ViewSpace, [E, Rest]) :-
    disconnected(ViewSpace, E),
    obstructed_r(ViewSpace, Rest).
```

The rule first collects all surrounding objects in the same room as the vantage point and object. It then generates the view-space and then recursively checks for obstructions over the view-space.

Figure 6.10: Visibility Example: Doorway and Stairs

The visibility attribute can be used to check if an architectural structure or interior design element is visible from a given viewpoint. For example, an architect might want to ensure that a staircase is visible from the main entrance of a building, because she doesn't want the staircase to go unnoticed. Figure 6.10 shows an example floor plan where the stairway is not visible from the doorway. In this example both the wall in red and column are obstructing the view from the doorway.

6.2 Architectural Concepts

The architectural concepts of privacy, continuity and enclosure describe experiential aspects of a design [29]. These terms are defined in the CRR in a hierarchical manner in which they are built using QSA and qualitative spatial relationships.

This section presents these three architectural concepts and demonstrates how each is defined in the CRR. Additionally, this section presents characteristics that are influenced by artefactual extension. As in the previous section, the purpose is not to investigate the correctness of the concepts or to attempt to compile a complete list; rather it is to look at how these concepts can be defined logically using architectural qualities and spatial / geometric primitives.

6.2.1 Privacy

In architecture, the concept of privacy implies a notion of separation and obscurity (not being visible). In the case of a room, privacy can be defined as being separated from the main entrance of the building and having doorways that enhances the room's privacy. A doorway can be defined as private with respect to its functionality of connecting the room to the main area of the house. Under this consideration, a doorway can enhance or hinder the privacy of a room by being visible or not visible from the main entrance.

Within the CRR, the concept of privacy has been formally defined for rooms. A room is private if it has the following attributes: (1) it is positioned on the opposing side of the building with regard to the main entrance and (2) all doorways into the room are not visible from the main entrance.

Privacy is represented in the CRR as a Prolog rule. The rule takes three arguments, which are the room's `crr_id`, the entrance's `crr_id` and the building's `crr_id`. The following is the Prolog rule for privacy.

```
private(RoomId, EntranceId, BuildingId) :-
    opposing_side(RoomId, EntranceId, BuildingId),
    get_doors_in_room(RoomId, Doors),
    not_visible_r(EntranceId, Doors).

not_visible_r(_, []).

not_visible_r(Id1, [Id2, Rest]) :-
    not_visible(Id1, Id2),
    not_visible_r(Id1, Rest).
```

Figure 6.11 demonstrates the privacy concept for two floor plans; Figure 6.11a shows a floor plan with a bedroom that is private, while Figure 6.11b shows a similar floor plan with a bedroom that is not private. The numbers represent the following privacy requirements: (1) the bedroom is positioned on the opposing side of the building with regards to the main entrance, and (2) the doorway is not visible from the main entrance.

Figure 6.11: Privacy

6.2.2 Continuity

In architecture, the concept of continuity describes a network of points that are mutually visible [28]. In the CRR, continuity is interpreted in the context of building navigation and wayfinding and is therefore dependent on continuity between doorways within the design. Other interpretations of continuity can use any element of the design (beyond doorways); however for the purposes here, continuity only deals with doorways. Under this interpretation a room is continuous if there is continuity between the doorways contained in it.

Continuity is defined as a network of doorways that are mutually visible. The partitioning of doorways that should be mutually visible is based on a localized containment hierarchy between rooms and doorways, in which doorways that are contained in the same room should maintain continuity. It does not make sense for doorways that are located on opposite sides of a building to be visible to each other, but doorways that are in the same room should be.

With respect to this definition, continuity is represented as a Prolog rule, which takes a `crr_id` for a room, and checks the visibility structures between each doorway located in the room. The following is the Prolog rule for continuity.

```
continuous(RoomId) :-
    get_doors_in_room(RoomId, Doors),
    mutually_visible_r(Doors).

mutually_visible_r(_, []).

mutually_visible_r(Id1, [Id2, Rest]):-
    mutually_visible_h(Id1, [Id2, Rest]),
    mutually_visible_r(Id2, Rest).

mutually_visible_h(_, []).

mutually_visible_h(Id1, [Id2, Rest]) :-
```

```

is_visible(Id1, Id2),
is_visible(Id2, Id1),
mutually_visible_h(Id1, Rest).

```

Figure 6.12 demonstrates the continuity concept for two floor plans. Figure 6.12a shows a configuration in which each door is mutually visible to the other doors, and therefore is continuous. Conversely, Figure 6.12b shows a configuration in which *door1* is not visible to *door2* and *door1* is not visible *door3*. The visibility between *door1* and *door2* is obstructed by a column and the visibility between *door1* and *door3* is obstructed by a wall. Architectural entities that are red indicate a visibility obstruction. This design is not continuous.

Figure 6.12: Continuity

6.2.3 Enclosure

In architecture, the concept of enclosure defines a space, as experienced from a given vantage point, as being confined or open. In the CRR, enclosure is defined by localized spatial structures in terms of their proximity to a given point. Under this interpretation, a room, apartment, or building can be experienced as both confined and open based on the location of the vantage point within the space.

Such a definition takes into account that enclosure is locally experienced. For example, an alcove in an otherwise open room feels confined when experienced from inside the alcove.

In the CRR, a point is defined as open if fewer than three walls or columns are near-to the point. Conversely, a point is defined as confined if three or more walls or columns are near-to the point. While this interpretation of enclosure is naive, it maintains the notion that proximity defines the spatial structures of openness and confinement in a localized environment.

Enclosure is implemented as two Prolog rules, one for confined spaces and one for open spaces. Both rules take a single argument that is a `crr_id` for the point being evaluated. The body of each rule calculates the distances characteristics of the local architectural entities to the point being evaluated.

```

confined_space(Id) :-
    room_containment(Id, Rm),
    get_entities_in_room(Rm, Entities),
    num_near(Id, Entities, N),
    N >= 3.

open_space(Id) :-
    room_containment(Id, Rm),
    get_entities_in_room(Rm, Entities),
    num_near(Id, Entities, N),
    N < 3.

num_near(_, [], 0).

num_near(Id1, [Id2|Rest], N) :-
    num_near(Id1, Rest, NRest),
    ( near(Id1, Id2) -> N is NRest + 1 ;
      N is NRest + 0 ).

```

Figure 6.13 shows a floor plan for a single room with various enclosure prop-

Figure 6.13: Enclosure, Confined and Open

erties. The room contains an alcove and a sequence of columns that will provide an environment of confinement. Points in between the columns and inside the alcove are confined because of the proximity of the walls and/or columns. The points outside of these areas are open because there are few structures nearby.

6.2.4 Artefactual Requirements

Artefactual requirements are a special case in this section because they are not strictly architectural concepts. However, they provide a unique set of architectural requirements that incorporate a breadth of design aspects. This section will take a look at two examples of artefactual interference.

Artefactual interference deals with the unintentional interactions of the operational and functional spaces of a building's structural form. For example, the operational space of a bathroom doorway that overlaps with the functional space of the bathroom sink will cause circulation problems into and out of the bathroom. In the CRR, this type of interaction can be detected using topological reasoning over the artefactual extensions of the entities.

Figure 6.14: Sink Doorway Interference

This type of artefactual interaction can be checked for in the CRR using a single rule that takes two arguments in the head of the rule. The body of the rule retrieves the artefactual geometry of each element and checks their topological relationship. It fails if the artefacts have an overlapping relationship.

```
bathroom_sink_door(sink_id, door_id) :-
    functional_representation(sink_id, SinkFunc),
    operational_representation(door_id, DoorOps),
    not_overlapping(SinkFunc, DoorOps).
```

Artefactual interactions can be between an architectural entity and artefactual extension. For example, the landing space (i.e., operational space) of a stair should be clear. Representing this requirement is similar to the bathroom sink example except that it checks for interactions between the stair's artefactual space with the design's physical space.

In the CRR this can be represented as a single rule. The body of the rule checks for an overlapping relationship between the landing space of the stair and physical structure of surrounding elements.

```
clear_stair_landing(stair_id, physical_id) :-
    operational_representation(stair, StairOps),
    physical_representation(physical_id, Physical),
    not_overlapping(StairOps, Physical)
```

6.3 Conceptual Requirement Validation

The CCR validates design requirements by querying the design's spatial representation using predicate calls to architectural concepts and QSA. Queries will return *true* if the design is consistent, or *fail* if the design is inconsistent per the query. A design is considered consistent if the design's spatial characteristics, as represented in its physical and functional geometries, matches with the definition of the design requirement specification. For example, a design is consistent per a design requirement of a window and doorway facing towards each other, if the spatial characteristic of the design, with regards to the window and doorway, match the spatial specification of facing towards each other. This requirement can be checked for using the following query given a door with *crr_id* *door543* and a window with *crr_id* of *window15*.

```
?- facing(door543, window15).
```

Queries to architectural concepts and QSA can be combined to build requirements. For example, a design requirement for a doorway and window being near to each other can be combined with the previous requirement of facing towards each other to build a new requirement. This requirement can be queried directly or implemented as a new Prolog rule.

```
?- facing(door543, window15),  
   near(door15, window15).
```

```
facing_and_near(Id1, Id1) :-  
    near(Id1, Id2),  
    facing(Id1, Id2).
```

Note that the direct query is specific to *door543* and *window15*, while the rule allows the user to input any *crr_id*'s into the query.

```
?- facing_and_near(desk52, door6).
```

Consider the design requirement for a bedroom being private. This requirement can be validated for a design by querying the privacy predicate, as demonstrated above. However, this definition of privacy might not be enough for a particular user. In the CRR, the user can easily extend the definition of privacy by building new Prolog rules that join additional architectural concepts, QSA, and spatial qualities to the existing definition. If the user wants the definition of privacy to include an additional requirement that the bedroom is far away from the living room, the user can construct a new rule by joining the privacy predicate with the proximity predicate. The following is the new predicate for privacy:

```
privacy_new(IdRoom, IdLivingRoom) :-  
    privacy(IdRoom),  
    far(IdRoom, IdLivingRoom).
```

Chapter 7

Museum Case Study

This chapter presents a case study that demonstrates the application of the CRR in the context of design for an art museum. The purpose is twofold: first, to examine design requirements of an art museum with regards to their effects on visitors' experiences in a museum; and second, to demonstrate the application of the CRR for validating these design requirements using a real world museum scenario.

This chapter is divided into two sections. The first section sets up the context of the case study by examining museum design characteristics that influence visitors' behavior and experiences within an art museum. This section has two parts. The first looks at museum characteristics that affect how visitors behave in museums galleries, and the second part looks at the characteristics of a museum's lobby that influence visitors orientation. The second section of this chapter demonstrates the CRR by applying the museum requirements to two real world museum designs.

7.1 Museums

Museums are a place of public education, in which art, historical artefacts, and scientific exhibits are put on display for the public to view [19]. While museums in general have many common threads in terms of their design requirements this case study takes the approach of narrowing the study to that of an art museum. This focuses the discussion to specific real world examples and maintains the perspective that this is not an exhaustive study of museum design requirements; rather it's an investigation into the nature and interpretation, with respect to the spatial characteristics of a set of design requirements. Additionally, the point is not to prove the validity of these requirements in an empirical sense; rather it is to show the process of representing and reasoning about design requirements in a formal computational framework.

The art museum is a unique and fruitful choice for this case study because it provides a rich set of design properties that have been formally studied and reported [31] [8] [19]. Within museum studies, visitor behaviour has been one of the primary focuses, starting with experiments dating back to the 1930's.

7.1.1 Museum Design Requirements

Pretend for a moment that you are an architect working on the initial design for a new modern art museum. During your first visit with the museum's curator you discuss several design requirements that are important for the new museum. Your discussion reveals that the curator wants the new art museum to adhere to the following two requirements:

1. Maximize visitor utilization of exhibitions

2. Maximize visitor orientation in lobby

While these requirements are abstract, they can be mapped to concrete spatial characteristics of museum design. The following section looks at several of these characteristics.

7.1.2 Maximizing Visitor Utilization of Exhibitions

This portion of the case study will look into four factors that influence visitors' movement patterns: positioning of doorways, spatial arrangement of display cases, positioning of furniture and statues, congestion, and exploration. These properties inform museum designers about the spatial characteristics that influence the utilization of the museum's exhibitions.

Positioning of doorways

In an important study [31], Melton discovered that the most influential factor to determine the amount of attention an exhibition piece receives is the pieces's position relative to doorways in a gallery room. Prior to this, museum researchers believed that the pieces's intrinsic properties of 'interestingness', color, and size were the contributing factors that determined the amount of attention a piece received. Melton explained why this happens; he demonstrated that the amount of attention a piece receives is directly related to the visitor circulation patterns within the museum and these circulation patterns emerge from the spatial characteristics of the museum's architecture.

Figure 7.1: Results from Melton's experiments on the effects of door positioning in an Art Museum

The crux of Melton's experiments suggested that visitors exit through the first door they encounter, which he referred to as the exit gradient. Figure 7.1 shows the results of an extensive study of visitors circulation patterns and stopping frequencies in an art museum. The study reported that almost 80 percent of visitors exited through the first exit they encountered before viewing the entire exhibition. Additionally, the study showed that the most frequently visited objects were those located along the shortest path from the entrance to the exit.

The results of Melton's studies inform architects of spatial characteristics that influence movement patterns in exhibitions. Specifically, doorways that are positioned on the same side of a gallery room and doorways that are near to each other cause visitors to pay attention to fewer exhibits than doorways that are positioned on opposing sides of a gallery room and doorways that are far away from each other.¹

Gallery Requirement 1: *Gallery doorways should be positioned on opposing sides of the gallery room.*

Gallery Requirement 2: *Gallery doorways should be far away from each other.*

¹These two requirements pertain to galleries with exactly two doorways. Galleries with more or fewer doorways have not been studied.

Figure 7.2 shows two gallery exhibition designs. Considering Gallery Requirement 1, Figure 7.2a shows a design that is consistent with the requirement, while Figure 7.2b shows a design that violates the requirement.

Figure 7.2: Positioning of Doorways in Galleries

Placement of display cases

The placement of display cases in galleries has been shown to influence visitors' movement patterns. Bitgood [9] showed in some environments that display cases that are located at the perimeter of a gallery generate the most amount of attention, while display cases that are isolated in the middle of the gallery received the lowest amount of attention.

Figure 7.3: Display case arrangements

Gallery Requirement 3: *Display cases should be located on the perimeter.*

Figure 7.3 shows two illustrations. Figure 7.3a show a display case that is isolated near the center of the gallery, while Figure 7.3b shows a display case that is located at the perimeter of the gallery.

Additionally, the placement of a sequence of display cases can disrupt circulation patterns when the sequence cuts horizontally across the view of a visitor as they are entering the room. This type of display pattern might cut off part of the gallery from the visitors normal visitation patterns by changing circulation pathways.

Gallery Requirement 4: *Display cases sequences should not cut horizontally across the view of a visitor located at an entrance.*

Figure 7.4: Display Case Sequence

Figure 7.1.2 shows two example galleries with displace case sequences. Figure 7.4a shows an example where the display case sequence does not cut horizontally across the visitor's view, while Figure 7.4b shows an example where the display cases sequence does cut horizontally.

Congestion

Congestion in museums can be caused by artefactual interference between the viewing space of displayed objects with movement pathways of visitors. In order

to reduce congestion in museums, designers should be aware of the placement of display cases and furniture (seating benches) within the gallery. The following requirement should be followed to reduce congestion in museums.

Gallery Requirement 5: *There should be no artefactual interference between museum display cases, museum seating and architectural entities.*

Positioning of benches and statues

The positioning and orientation of benches and statues can increase the amount of attention an exhibition receives and change circulation patterns within a gallery [38] [1]. Studies have shown that seating facing a doorway or window will decrease attention in the Gallery.

Gallery Requirement 6: *Gallery seating should face away from doorways and windows.*

The orientation of a statue to doorways can increase the attention the statue receives as well as objects close to it. Statues that face away from doorways encourage visitors to move around the statues in order to get a better view.

Gallery Requirement 7: *Statues should face away from doorways.*

Exploration

The way visitors explore an exhibition is influenced by the spatial characteristics of the layout, as well as the characteristics of each gallery room. Layouts that are sequential (one way in, one way out), promote a controlled and orderly flow of movement. Museums of history and science are well served by this type of design

because they have a narrative to tell that is chronological. On the other hand, layouts that present the visitor with path choices (multiple doorways), promote free-flowing exploration. In this type of layout it is important that visitors are aware of the path choices that are available. This means that doorways are visible between each other throughout the layout. This concept of visibility between a network of points is referred to as continuity and is an important characteristic of layouts that present visitors with multiple path choices.

Gallery Requirement 8: *The layout of the museum should maintain continuity between doorways within each gallery room.*

7.1.3 Visitor Orientation in the lobby

The lobby is the first place visitors encounter when they enter a museum. The design of the lobby is important because it not only provides a first glimpse of the museum's exhibits but it is the place where the museum provides services to its visitors, including: a ticketing desk, bathrooms, and entrances to exhibitions [8]. The aim of the requirements for the lobby focus on making it easy for visitors to locate services and orient themselves to the new environment.

Upon entering the lobby most visitors need to find the ticket area. It is important, therefore, that the ticketing area is visible from the entrance. Additionally, studies have shown that people first look to their right hand side when entering a building. By placing the ticket area on the right hand side of the lobby, visitors are more likely to notice it immediately upon entering the lobby. Finally, the ticketing area should be close to visitors when they enter because it is likely one of the first places they will need to visit.

Lobby Requirement 1: *The ticketing area should be visible from the main entrance.*

Lobby Requirement 2: *The ticketing area should be on the right hand side of the lobby from the entrance.*

Lobby Requirement 3: *The ticketing area should be close to the entrance.*

After obtaining a ticket most visitors will try to locate the entrance to the exhibitions. Therefore, exhibition entrances should be visible from the ticketing area to make it easy for visitors to find where they need to go next.

Lobby Requirement 4: *Exhibit entrances should be visible from the ticketing area.*

7.2 Application of CRR for Museum Design

This section presents the application of the CRR to two real world museums using the gallery and lobby requirements, and is organized as follows. The first part introduces the two museums that are used in the case study and briefly describes the process of importing them into the CRR from ArchiCAD. The second part presents the results of using the CRR for validating the museum gallery and lobby requirements to the two museums.

7.2.1 Museum Designs and Import

This case study uses the Asian Art Wing of the Philadelphia Museum of Art [35] and the lobby of the Museu Calouste Gulbenkian [12]. The Asian Wing of

the Philadelphia Museum of Art will be used for the demonstration of the gallery requirements, while the lobby of the Museu Calouste Gulbenkian will be used for the demonstration of the lobby requirements. The motivation behind using real world museums instead of contrived examples is to demonstrate the CRR in a real world context with regards to the practicality of museum design requirements.

Figure 7.5: Philadelphia Museum of Art

The Asian Art Wing (gallery rooms 226 and 238-245), shown in Figure 7.5b, and the lobby of the Museu Calouste Gulbenkian were manually built in ArchiCAD 14. The dimension and scale of the rooms were obtained from the museums to ensure as much accuracy to the real design as possible. Benches, display cases and statues were added to demonstrate some of the museum requirements. Figures 7.7 and 7.6 show the floor plans for both museums in ArchiCAD.

Figure 7.6: Lobby of Museu Calouste Gulbenkian

Figure 7.7: Asian Wing of Philadelphia Museum of Art

Once the museums are built in ArchiCAD they are imported into the CRR via a Multi-Modal Spatial Data Access Framework [37]. This framework provides the CRR with access to various design information, via the Industry Foundation Class data exchange format [22], including: the physical representation for each architectural entity, and room / building / story containment relationships. Arte-

factual representations, room to wall boundary relationships, and wall to door / window containment relationships are manually added into the CRR. Future versions of the Multi-Modal Spatial Data Access Framework plan on supporting these informational design aspects. The import process consists of a parser written in Python that parses a Design Space schematized XML file, produced by the Multi-Modal Spatial Data Access Framework, and outputs a Prolog representation of the design that is used by the CRR.

7.2.2 Validating Museum Requirements

The results of applying the museum gallery and lobby requirements to the Philadelphia Museum of Art and the Museu Calouste Gulbenkian are presented in Table 7.1. This table contains three columns. The first column identifies the museum requirement, the second column shows the CRR query used to check for the requirement, and the third column show all violations that were discovered by the CRR. The results here demonstrate two outcomes. The application of the gallery requirements to the Philadelphia Art Museum gives an illustration of where the design is inconsistent with regards to the requirements, while the application of the lobby requirements to the Museu Calouste Gulbenkian gives an illustration where the design is consistent with the requirements.

Requirement violations reported in Table 7.1 can be visually verified by looking at Figure 7.7. For example, Gallery Requirement 1, which specifies that doorways should be on opposing sides of a gallery room, can be visually inspected to verify the CRR results. In this example, the CRR reported that the following doorways and gallery rooms are in violations of Gallery Requirement 1:

- door4, door10, gallery240

Req.	CRR Query	Violations
GR1	<code>opposing_side(door_id, door_id, room_id)</code>	<ul style="list-style-type: none"> ◦ door4, door10, gallery240 ◦ door5, door11, gallery243 ◦ door7, door12, gallery241
GR2	<code>far(door_id, door_id)</code>	<ul style="list-style-type: none"> ◦ door4, door10 ◦ door5, door11 ◦ door7, door12
GR3	<code>near(display_id, wall_id)</code>	◦ display4
GR4	<code>horizontal(display_list)</code>	◦ (display0, display1, display2, display3)
GR5	<code>artefact_interference(id, id)</code>	◦ door8, display8
GR6	<code>facing_away(bench_id, door_id)</code>	<ul style="list-style-type: none"> ◦ bench0, door6 ◦ bench1, door3
GR7	<code>facing_away(statue_id, door_id)</code>	◦ statue0, door6
GR8	<code>continuity(room_id)</code>	◦ gallery239
LR1	<code>visibility(entrance_id, desk_id)</code>	no violations
LR2	<code>right_side(entrance_id, desk_id)</code>	no violations
LR3	<code>near(entrance_id, desk_id)</code>	no violations
LR4	<code>visibility(desk_id, door_id)</code>	no violations

Table 7.1: Case Study Results

- door5, door11, gallery243
- door7, door12, gallery241

Visual inspection of Figure 7.7 confirms that these doorways are *not* on opposing sides of the gallery room and therefore are in violation of Gallery Requirement 1. As another example, Gallery Requirement 8 specifies that every Gallery should maintain continuity between doorways. The results of this case study shows that *gallery 239* is in violation of the requirement. The cause of this violation is because *door3* and *door10* are not visible from *door9*, and vice versa. Figure 7.8 shows a 3-D view of *gallery 239* from the perspective *door10* (this graphic was rendered in ArchiCAD) and shows that in fact *door3* and *door10* are not visible from *door9* because there is a wall obstructing the view.

The process of manual verification can be used to inspect the other gallery

Figure 7.8: Visibility in Gallery239

requirement violations reported by the CRR. The result of doing so will show that the CRR accurately reported violations for this case study. To illustrate this, Figure 7.9 shows the Philadelphia Art Museum floor plan with labels for corresponding gallery requirement violations as reported in Table 7.1.

Similarly, this same process can be used to verify that the lobby of the Museu Calouste Gulbenkian is consistent with the lobby requirements. For example, Lobby Requirement 2 specifies that the ticketing desk should be on the right hand side of the entrance from the perspective of entering the museum. Inspecting Figure 7.6 shows that in fact the ticketing desk is on the right hand side. Manual inspection of the other lobby requirements shows that the Museu Calouste Gulbenkian is consistent with the lobby requirements as reported by the CRR.

This case study demonstrated the application of the CRR to reason about conceptual design requirements given an underlying CAAD-based spatial representation of the design. The crux of the work here is the use of a qualitative

Figure 7.9: Gallery Requirement Violations

spatial layer that intermediates between the conceptual level and quantitative level. The results show that the CRR is capable of accurately detecting design requirement violations and informs the designer when a design is consistent or inconsistent per the designs requirements.

7.2.3 Discussion of Museum Requirements

Validating the requirements in Table 7.1 presented a variety of difficulties from both a user and computational perspective. This subsection will examine these difficulties and provide some insight into the process of validating design requirements.

From a user perspective the biggest difficulty in using the CRR is handling

the quantification of a requirement. This means that the user is responsible for keeping track of the results of previous queries if necessary. For example, Gallery Requirement 4 (GR4) specifies that display cases should not be positioned as an island. In other words this requirement can be stated as, for every display case there should be at least one wall that it is near. For this requirement to be validated, the user must check the proximity relationships between each display case and walls surrounding it, while keeping track that the display case is near to at least one wall. In the case that no walls are near to a display case, it is considered to be an island.

In the museum case study presented here, *display4* is in violation of this requirement (GR4) because it is not near to any surrounding walls. To come to this conclusion in the case study there are four proximity queries that need to be checked before the requirement can be verified as a validation. Each query is of the form, *near(display4, wall_id)*, where *wall_id* is the *crr_id* for one of the wall surrounding the display case. If all of the queries return with a *fail* then the display case is in violation of the requirement.

Not all requirements requires the user to keep track of the results of previous queries. For example, Gallery Requirement 6 (GR6) involves checking a single orientational relationship between a doorway and a bench. For this requirement the user only needs a single query to check the facing relationship between a doorway and a bench. In the museum case study, *bench0* and *door6* are in violation of this requirement. This can be checked with a single query, *facing-away(bench0, door6)*.

From a computational perspective the complexity of each requirement can vary. For example in Museum Requirement 1 the CRR is queried for a single orientational relationships. - req 1 - one orientational relationships - req 8 -

network of visibility relationships that are a series of topological relationships.

It should be noted that the severity of a requirement violation are not accounted for in the CRR.

Chapter 8

Conclusions & Future Work

This thesis addressed the problem of validating conceptual design requirements given a CAAD-based spatial representation. The main objectives towards accomplishing this goal were identified as:

- the identification, formal modeling, and spatial interpretation of architectural concepts
- the qualification of CAAD-based spatial models, and
- the use of qualitative spatial representation and reasoning techniques to link high level concepts to low level spatial quantities

To accomplish these objectives the CRR was implemented as a framework for validating conceptual design requirements against a CAAD-based spatial representation. The development of the CRR was based on a design hierarchy that maps architectural concepts and QSA at the design level to spatial patterns at the quality level and links, via qualification, spatial quantities at quantity level to qualitative spatial relationships in the quality level. The main contributions

of this thesis are:

- the implementation of a framework that validates conceptual design requirements against a CAAD-based spatial representation
- the identification of a set of QSA and architectural concepts
- the development of a multi-hierarchy that maps QSA and architectural concepts to spatial patterns in the qualitative level
- the implementation of spatial constraint solvers that qualify CAAD-based spatial models
- the development of a real world museum case study that demonstrates the applicability and effectiveness of the CRR in validating museum design requirements

8.1 Future Work

While this thesis addressed several key issues in the area of conceptual requirements reasoning, there are several areas where future work can be done.

Automatic generation of artefactual geometry

In the current version of the CRR artefactual geometries are manually added to the design representation. This limitation means that artefactual reasoning can not happen automatically over a CAAD-based spatial representation. Future work in the area of automatically generating artefactual geometries from a CAAD-based spatial representation will be beneficial to the usability of the CRR.

Quantification of design requirements

The current version of the CRR does not handle queries involving quantification over the design. This means that each query is isolated to a specific instance in the design. Whereas a design requirement specifies that all doorways must face away from all windows in the same room, the user is responsible for checking each instance of doorway and window in the design. This requires the user to keep track of the results of previous queries in order to validate certain requirements. The ability to have the CRR automatically provide quantification over design requirements would allow the user to focus more on requirements reasoning and less on the book keeping of results.

Data model for conceptual architectural design reasoning

The architectural representation model in the CRR is based on room containment relationships. Future work is needed to investigate the best data structures and representation models for reasoning about architectural design at a conceptual level. To this end two issues are identified here. 1) The containment relationship of walls within a building context is still unclear. The question remains if walls should be contained in a room or if they are boundaries to rooms. In general the entire hierarchical containment structure of an architectural design needs to be studied. 2) The artefactual geometries for doorways are usually contained in two separate rooms. The question remains how this situation should be represented to effectively reason about artefactual requirements.

Benchmark of spatial constraint solvers

An evaluation of the spatial constraint solvers in terms of handling large number of queries will be an important aspect in expanding the CRR to larger data sets. Currently, the spatial constraint solvers handle only a few (< 10) queries at a time and therefore respond instantaneously. Expanding the CRR to handle larger data sets and more complex spatial queries will require an investigation into how well the spatial constraint solvers scale.

Bibliography

- [1] P. Adams, D. Allan, P. Coremans, H. Daifuku, M. Harrison, B. Molajoli, and P. Schommer. *The Organization of Museums: practical advice*. United Nations, 1960.
- [2] O. Akin, R. Sen, M. Donia, and Y. Zhang. Seed-pro: Computer-assisted architectural programming in seed. *Journal of Architectural Engineering*, 1(4), 1995.
- [3] C. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.
- [4] J. Almendros-Jiménez. Constraint logic programming over sets of spatial objects. In *Proceedings of the 2005 ACM SIGPLAN workshop on Curry and functional logic programming, WCFLP '05*, pages 32–42, New York, NY, USA, 2005. ACM.
- [5] N. Bayazit. Investigating Design: A Review of Forty Years of Design Research. *Design Issues*, 20:16–29, 2004.
- [6] M. Bhatt, F. Dylla, and J. Hois. Spatio-terminological inference for the design of ambient environments. *Conference on Spatial Information Theory (COSIT'09)*, pages 371–391, 2009.

- [7] M. Bhatt and C. Freksa. Spatial computing for design: An artificial intelligence perspective. *SF International Workshop on Studying Visual and Spatial Reasoning for Design Creativity SDC'10*, 2010.
- [8] S. Bitgood. Environmental psychology in museums, zoos, and other exhibition centers. In *Handbook of Environmental Psychology*, pages 461–480, 2002.
- [9] S. Bitgood, W. Hines, and W. Ford. Visitor circulation through a changing exhibits gallery. In *In Visitor Studies: Theory, Research, and Practice*, 1992.
- [10] A. Borrmann, J. Hyvarinen, and E. Rank. Spatial constraints in collaborative design processes. *Proc. of the Int. Conf. on Intelligent Computing in Engineering (ICE'09)*, 2009.
- [11] C. Caldern, M. Cavazza, and D. Diaz. Clp a technology for the interactive resolution of spatial configuration tasks in a virtual environment. *Special Issue Architectural informatics*, 2006.
- [12] Calouste Gulbenkian Foundation. Museum Calouste Gulbenkian, 2010. <http://www.museu.gulbenkian.pt/main.asp?lang=pt>.
- [13] H. W. Chun and E. Ming-Kit Lai. Intelligent critic system for architectural design. *IEEE Trans. on Knowl. and Data Eng.*, 9(4):625–639, 1997.
- [14] A. Cohn and S. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundam. Inf.*, 46:1–29, January 2001.
- [15] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

- [16] Design Space, 2010. <http://www.sfbtr8.uni-bremen.de/designspace.html>.
- [17] C. Eastman. Invited keynote presentation, new methods of architecture and building. 2004.
- [18] C. Eastman, R. Sacks, and G. Lee. Functional modeling in parametric cad systems.
- [19] J. Falk and L. Dierking. *The Museum Experience*. Howells House, 1948.
- [20] C. Freksa. Qualitative Spatial Reasoning. *Cognitive and Linguistic Aspects of Geographic Space*, pages 361–372, 1991.
- [21] C. Freksa. Using orientation information for qualitative spatial reasoning, 1992.
- [22] T. Froese, F. Grobler, J. Ritzenthaler, K. Yu, and S. Staub. Industry foundation classes for project management - a trial implementation, 1999.
- [23] T. W. Frühwirth. Constraint handling rules. In *Constraint Programming*, pages 90–107, 1994.
- [24] T. W. Frühwirth. Theory and practice of constraint handling rules. *The Journal of Logic Programming*, 37(1-3):95 – 138, 1998.
- [25] M. Gavanelli and F. Rossi. Constraint logic programming. In *25 Years GULP*, pages 64–86, 2010.
- [26] Graphisoft Inc. ArchiCAD 14, 2010. <http://www.graphisoft.com/>.
- [27] J. Jaffar and J. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '87, pages 111–119, New York, NY, USA, 1987. ACM.

- [28] S. Key. A computable language of architecture: Description of descriptor language in supporting compound definitions. *Design Tool Development eCAADe 27*, 2009.
- [29] K. Koile. An intelligent assistant for conceptual design: Informed search using a mapping of abstract qualities to physical form, 2004.
- [30] G. Korte. *The GIS Book*. ONe Word Press, Albany, New York, USA, 2001.
- [31] A. Melton. *Problems of Installation in Museums of Art*. 1935.
- [32] H. J. Miller and J. Han. *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis, Inc., Bristol, PA, USA, 2001.
- [33] R. Moratz. Representing relative direction as a binary relation of oriented points. In *Proceeding of the 2006 conference on ECAI 2006*, pages 407–411, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.
- [34] L. Myers and J. Pohl. Icads expert design advisor: an aid to reflective thinking. *Knowledge-Based Systems*, 5(1):41 – 54, 1992. Artificial Intelligence in Design Conference 1991 Special Issue.
- [35] Philadelphia Museum of Art, 2010. <http://www.philamuseum.org/>.
- [36] D. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In *Proceedings 3rd International Conference of Knowledge Representation and Reasoning*, 1992.
- [37] C. Schultz and M. Bhatt. A multi-modal data access framework for spatial assistance systems: use-cases with the building information model (bim/ifc). In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on*

- Indoor Spatial Awareness*, ISA '10, pages 39–46, New York, NY, USA, 2010. ACM.
- [38] G. Stavroulaki and J. Peponis. The spatial construction of seeing at castelveccchio. In *Proceedings of the 4th International Space Syntax Symposium*, pages 339–346, 2003.
 - [39] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, Cambridge (MA), 1986.
 - [40] L. Sullivan. The tall office building artistically considered. *Lippincott's Magazine*, 1896.
 - [41] C. Tostoes, A. Carapinha, and P. Corte-Real. *Gulbenkian: Architecture and Landscape*. Calouste Gulbenkian Foundation.
 - [42] M. Wallace. Constraint logic programming. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2407 of *Lecture Notes in Computer Science*, pages 35–64. Springer Berlin / Heidelberg, 2002.

Appendix A

Appendix

A.1 Spatial Abstraction of Architectural Entities

This appendix shows how architectural entities and interior designs objects are spatially abstracted into points, directed points, and convex hulls. Table [A.1](#) indicates which architectural entities and interior design objects can be abstracted to which spatial abstractions.

architectural entity	point	directed point	convex hull
doorway	X	X	X
window	X	X	X
wall	X		X
column	X		X
space	X		X
interior design object	X	X	X

Table A.1: Spatial Abstractions

Figure A.1: Doorway Point Abstraction

Point

All architectural entities and interior design objects have a point abstraction that is defined as the centroid of the entity's physical representation. Figure A.1 shows the point abstraction for a doorway. Note that the point is the centroid of the doorway's physical representation. The point abstraction is commonly used in orientational reasoning and therefore is an approximation of the object's location in the design. The following Prolog code shows the transformation rule for point abstractions.

```
point_abstraction(Id, Pt) :-
    physical_representation(Id, P),
    centroid(P, Pt).
```

Directed Point

Doorways, windows, and some interior design objects have a directed point abstraction. A directed point is represented as a tuple $(point, direction)$. The point is defined as the centroid of the physical representation, similar to the point abstraction. For interior design objects the direction is defined by the user, while for doorways and windows it is defined by an intrinsic spatial property of the doorway or window.

Figure A.2: Doorway Point Abstraction

For doorways and windows direction is based on the interpretation of the nature of the entity within the context of a design. In other words, the direction is interpreted as a vector that emanates from an entity's point abstraction towards the room it is contained in. Specifically, direction is defined as the perpendicular vector to the intrinsic x-axis of the object. Figure A.1 shows the directed point abstraction for a doorway. Note that the direction is perpendicular to the x-axis of the doorway's physical representation. Based on this definition there will be two perpendicular lines that define the direction. In this situation, the context of the space in which the entity is located is needed to decide which of the two directions to use.

```
dir_point_abstraction(Id, (Pt, Dir), Rm) :-
    physical_representation(Id, P),
    centroid(P, Pt),
    direction(P, Pt, Rm, Dir).
```

Convex Hull

All architectural entities have a convex hull abstraction. The convex hull is calculated over the physical representation using the divide and conquer convex hull algorithm [15]. The result of the algorithm is a convex hull representation

that is a list of points sorted in counterclockwise order. Convex hulls are commonly used in topological reasoning.

```
convex_hull_abstraction(Id, Convex) :-  
    physical_representation(Id, P),  
    convex_hull(P, Convex).
```