A Unified Semantics Space Model*

Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon

System Research Group, School of Computer Science and Informatics,
UCD, Dublin, Ireland
juan.ye@ucd.ie

Abstract. Location-aware systems provide customised services or applications according to users' locations. While much research has been carried out in developing models to represent location information and spatial relationships, it is usually limited to modelling simple environments (cf. [13, 19, 3]). This paper proposes a unified space model for more complex environments (e.g., city plan or forest). This space model provides a flexible, expressive, and powerful spatial representation. It also proposes a new data structure – an integrated lattice and graph model – to express comprehensive spatial relationships. This structure not only provides multiple graphs at different abstraction levels, but it also collapses the whole map into smaller local graphs. This mechanism is beneficial in reducing the complexity of creating and maintaining a map and improving the efficiency of path finding algorithms.

1 Introduction

The development of location-aware systems has become commonplace recently. This was encouraged by the availability of numerous available location sensing devices [11] and by a popular demand for location-aware applications. A huge number of location models have been developed – however most of them tend to either service particular sensing abilities or application requirements.

Consider a complex real-world environment, such as a city or forest. It can be partitioned through multiple hierarchies (e.g., postcode areas, districts or compass directions) and involve a huge number of places (e.g., hundreds of streets or thousands of buildings). In this environment, it may be necessary to describe the location of a certain entity in various ways depending on the available location sensors. Most of existing space models only provide traditional types of spatial representations, such as symbolic and geometric representations. Especially some of these models support a single coordinate reference system, because they only have one or two precise sensors that provide location data in a coordinate format. As such, the ability of these models to flexibly express location information is limited.

Our space model aims to support a powerful and expressive spatial representation. It absorbs the best practices from existing models so as to support the traditional spatial

* This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 "Towards a semantics of pervasive computing" and 04/RPI/1544 "Secure and predictable pervasive computing".

representations, and makes improvements over them. This model supports multiple coordinate systems from two perspectives: different global coordinate systems to support different sensing technologies; and user-defined local coordinate systems to support customised representations of environment instead of forcing all the spatial representations in a uniform coordinate system. Furthermore, this space model also supports *relative* location representation [15]. Relative locations are necessary when the location cannot be exactly specified or defined, or if the location is dynamic or moving. For example, 'a place 500m east of this building", or "I am in the canteen of this train". Our approach can flexibly define a local reference system to describe these locations in different directions by combining various kinds of spatial representations, while not being limited to coordinates.

In complex environments it may also be necessary to construct a detailed map that can provide a path to a destination for an entity among a large number of places (with varied levels of granularity). There are two underlying models to organise spaces: hierarchical and graph models, which represent *containment* and *connectedness* relationships respectively. The typical approach to constructing a location map is to build a single huge graph for the whole environment, while fixed at a certain granularity (e.g., room-leveled spaces). This graph cannot be flexibly extended into coarser-grained spaces (e.g., buildings or streets), or into finer-grained spaces (e.g., desks). Although additional graphs may be built for these spaces, it is complicated to build a mapping between them or coordinate them in applications. Most research experiments usually take place in a building, so a single graph is easy to build and maintain. However, a single graph for larger-scaled environments will take much effort and time to build and to maintain its consistency and integrity. Also, existing models do not provide an approach to collapse this graph so as to reduce the construction complexity.

In our space model, we propose a new data structure - a lattice integrated with graphs to represent spatial relationships for complex environments. This space model applies the lattice model to represent the *containment* relationship and applies the graph model to represent the *adjacency* and *connectedness* relationships between spaces at the same abstraction level. The integrated model builds a lattice model for all the spaces under a certain partition approach. The graph models are embedded in the lattice model where each node is associated with a graph whose vertices are the immediate sub spaces of the node and whose edges are the adjacency and connectedness relations between these vertices.

With this space model, system designers can build a single model to express all the spatial relationships at once. They do not need to maintain separate hierarchy and graph models and mappings between them. Furthermore, users can build graph models at different abstraction levels whose hierarchies are managed in a lattice model. Even at a certain abstraction level, the whole graph can be further divided into smaller graphs that are associated with sibling nodes in the lattice. That is, each sibling node manages a local graph that is part of the whole map. These local graphs can also be integrated to form the whole map through a special space – a *sensitive space* – in the lattice model. For a given set of spaces, a sensitive space is the largest of their common sub spaces, whose detailed discussion will be in Section 3. This approach makes the graph model easy to create and maintain. In addition, our space model improves the efficiency of path

finding algorithms. In existing models, the path finding algorithm usually works on a large amount of location data. Heuristic approaches are applied in the algorithm. Also, our space model has the ability to reduce the initial searching space for path finding, which potentially improves search efficiency.

This paper is organised as follows. Section 2 reviews the mature location models and distills the best practices from them. Section 3 introduces our space model in representing location information and spatial relationships. Section 4 will demonstrate the implementation of relative location representations and the path finding based on the integrated model. Finally, in Section 5 we conclude the paper and point to future research problems in this work.

2 State of the Art of Location Models

This section will review the main types of location models including geometric, symbolic, hybrid, and semantic models. Their novel ideas and techniques will be introduced at the aspects of representing location information and spatial relationships.

At the aspect of representing location information, Jiang's model [13] proposes a detailed method to represent geometric location in multiple coordinate reference systems. A sub space's coordinate system is specified by defining its *origin point* and *axes* within its super space's coordinate system. The origin point is specified as a displacement vector in the super space's coordinate system, while axes are specified as three unit vectors in the form of a matrix, called *rotation matrix*. A simple linear algebra is applied to convert coordinates under different reference systems. This approach is built on a sound mathematical foundation. However, a sub space's coordinate system is defined only under its super space's coordinate system, which is too restricted and not flexible. Our space model will borrow the basic idea of this approach and make improvement on the flexibility of defining local coordinate systems.

When it comes to describing the shape of a three-dimensioned space, Coschurba's model [6] proposes a 2.5-dimensioned approach to describe a three-dimensioned (3D) shape by specifying its base as a two-dimensioned (2D) and its height as a numeric value (0.5D). Only the coordinates for the space's base shape are recorded, which can reduce the amount of coordinate data and allow applying geometric computations on each shape. This approach for defining shapes and representing coordinates is borrowed to our space model and is further extended to more shapes.

Korkea-aho [15] introduces a common data set and an extensible framework of expressing location information in the Internet. He proposes a relative location as a specific type of descriptive location, that is, the location of an object is described relative to some other objects, such as, "10 meters North to the shop", or "the area centered around the building in a 100-meter arm". The relative location representation is very useful, and is not introduced in existing location models. Our space model will extend the spatial representations to cover this relatively spatial expression.

The HP Cooltown project [18] introduces a semantic representation, which is orthogonal to symbolic and geometric representations. The semantic representation provides other information around its place, such as a bus route or a snapshot of interest. In our space model, a space object may describe the spatial features corresponding to the

location of an entity, but other non-spatial features will necessarily be assigned to the corresponding entity object. For example, a building entity has its location described by a space object, whose symbolic representation is "science building". The building entity can be extended with other non-location contextual information including its functions, reception, activities, or services. The information expressed in a semantic representation of the Cooltown project is part of this extended contextual information.

At the aspect of representing spatial relationships, Becker [2] has analysed and summarised different types of symbolic location models: set-based, hierarchical, and graph-based models.

A set-based model organises symbolic locations into sets according to the spatial containment relationship. A typical example of the set-based model is the EasyLiving project at Microsoft [4]. It applies simple, computable set operators to evaluate the overlapping relationship, and to compare non-quantitative distances. A set-based model also can express the connectedness relationship by defining *neighbourhood* sets into pairs of directly connected locations. Set-based models can provide an explicit semantics of containment and overlapping relationships; however, the connectedness relationship results in tremendous number of sets and the notion of distance is not quantitatively computable.

A hierarchical model is a special case of set-based models. Similarly, a hierarchical model can express the explicit containment relationship, and it fails in supporting the connectedness relationship and the quantitative notion of distance. It is a most popular model applied in current location models such as Jiang's model [13], Schmidt's model [20], Durr's model [8], and MiddleWhere [19]. There are two hierarchical structures: a tree to organise non-overlapping locations, and a lattice to organise overlapping locations. Compared to a set-based model, a hierarchical model can represent the structure of locations more explicit and more intuitive. Moreover, these characterised structures support more efficient and optimised computations, such as a traversing algorithm in a tree, and the join and meet operations in a lattice. Compared to a tree model, the lattice model has more flexible expressivity and has been widely applied to location models such as Kainz' model [14], the Geocost model [6], and MiddleWhere [19].

A graph-based model is applied to express the connectedness relationship and the notion of distance. Hu's model [12] proposes a semantics graph model called *exit hierarchy*. In a graph, each node represents a symbolic location, and each edge represents the connectedness relationship, which quantitative distance can also be assigned on as a weight. Furthermore, nodes or edges can be attributed to more contextual information, such as passability limits, or transportation restrictions. A graph-based model provides an explicit and extensible semantics for the connectedness relationship and a quantitative notion of distance. It benefits in navigating an environment and answering the shortest path queries. However, its deficiency exists in the limited description level of locations (e.g., at the room level), and in the difficulty of expressing the containment relationship.

From the above analysis, none of the three models can have the adequate capability of representing all the spatial relationships. The set-based and hierarchical models are good at expressing containment and overlapping relationships, while the graph-based model is good at expressing connectedness relationships and the quantitative notion of

distance. We take all these best practices and improve them in our unified formal model of space.

3 A Unified Formal Model of Space

This section will build a formal semantics to richly express and model spatial information and relationships. The space model can provide multiple representations for any space in reality. It also can describe various spatial relationships, including containment, adjacency, connectedness, disjointness, overlapping, and distance.

3.1 Representation of Spatial Information

There are diverse ways to represent location information, however, the elementary types of spatial representations are geometric and symbolic representations. A *geometric representation* characterises a space with its geometric shapes and a set of coordinate points under a certain Cartesian Coordinate Reference System (CCRS). A *symbolic representation* characterises a space with a human-friendly descriptive label. A *hybrid representation* is introduced, if a space has a symbolic and geometric representation, both of which map to each other.

The space model supports a resolute or relative representation (Figure 1). A resolute representation can be a geometric, symbolic, or hybrid representation of the former two types. A resolute representation is usually used to describe an explicit space, such as "a conference room", or a coordinate $[4.50, 2.09, 3.87]$. A relative representation is used to represent an implicit space by specifying a base and an offset representation, both of which are directly or indirectly related to a resolute representation. In the following, we will detail the resolute representation first.

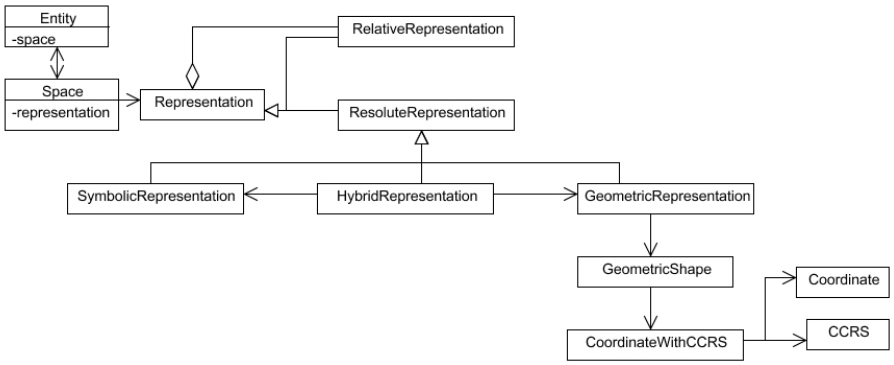

Fig. 1. Space Representation

A space is physically featured by different geometric shapes. A geometric shape can be regular, which is portrayed through a mathematically computable way. A regular

shape can be a primitive 2D or 3D shape, or a composite shape of primitive shapes. A few of primitive shapes have been listed, such as point, line segment, polygon, and sphere. If a shape cannot be depicted using a regular shape, then it is regarded as irregular.

The shape structure can be extensible to more sophisticated shapes in particular applications. For each space, coordinates are chosen and organised in a characteristic format in terms of the space's geometric shape. Typical shapes and their associated coordinate formats are listed as follows.

- *Point*: one coordinate;
- *Line segment*: two coordinates for two ends of a line segment;
- *Polygon*: a set of coordinates for each vertex of a polygon; especially, *Rectangle* is described by a pair of coordinates for diagonal points;
- *Circle or sphere*: a coordinate for a center and a numeric value for its radius;
- *Cube*: a set of coordinates for its base polygon and a numeric value for its height.
- and so on.

Coordinates for a composite shape can be obtained by integrating characteristic coordinates from the primitive shapes that it is composed of. However, an irregular shape cannot be easily described. A simplified solution is either to convert an irregular shape to a set of similar regular shapes or to pick up a few characterised coordinate points.

The coordinate representation is closely related to the geometric shape of a space. This approach can reduce the number of coordinate data and help organise them in a computable way. It is easy to do some spatial computations, such as computation of area or volume, and evaluation of whether a coordinate is in a space or not. For example, if a coordinate point and a sphere space are given, the evaluation can be computed by comparing its radius with the distance between this point and its center. If the distance is longer than the radius, then this point is considered out of the sphere-shaped space.

The space model supports multiple Cartesian Coordinate Reference Systems (CCRS), including global (such as WGS84, Gauss-Kruger, and UTM) and user-defined coordinate reference systems. The global system is applied according to different positioning sensors. For example, GPS provides coordinates under WGS84. Local reference systems are applied in terms of sensors and customised representation of an environment or a space. The Ubisense provides coordinates under a user-defined CCRS at the beginning when it is set up and configured. In a local environment, especially a dynamical environment, a user-defined CCRS is usually applied, when only relative location information is required. For example, in a train or ship, only relative location information in the train space is concerned, so a CCRS can be created locally in a train.

This space model supports multiple CCRSs and conversion of coordinates between these different CCRSs. Each space can be described with a set of coordinate sets, and each coordinate set is defined under a certain CCRS. A local CCRS is defined by specifying its origin point and axes, both of which can be referred to any existing CCRS, not necessarily its super space's CCRS. A default CCRS is defined as a primitive reference system that can be directly or indirectly referred to by all the other CCRSs. The conversion techniques has been borrowed from Jiang's model [13].

A symbolic representation for a space is a human-friendly, string-based descriptive label. A label can be a real name of a space like country's name, city's name, university's name, and so on. A space can be labeled in terms of its functionality like "conference room", "foyer", and "registration office ", etc. A space can also be named with its owners like "Waldo's house", "secretary's office" and so on. If a space has geometric and symbolic representations, then a bi-literal link can be built between them. For spatial computation, the geometric properties can be obtained from a symbolic representation; for expression, a symbolic label can be acquired according to the corresponding geometric representation.

So far, we have discussed the resolute representation for explicit location information, and now we will introduce a relative representation. A relative representation consists of a base and offset representation. Both base and offset representations can be any type of representation, and they are associated by a certain structural relation.

An offset representation can be adjacent to or centered around a base representation. If it is centered around a base representation and is specified in a given distance, a relative representation can be expressed as a circle (or sphere) by taking the base one as an origin point, and the distance as a radius (Figure 2). If it is adjacent to a base representation and is specified with a distance to the base and a list of degree to each direction. The direction is a standard compass direction description, including "East", "West", "South", "North", "Upper", and "Lower". The relative representation is defined in a similar approach of a local CCRS. This local CCRS is specified by taking the base location as an origin point, and taking the compass direction as the axes' direction. A relative representation will be located by projecting the associated distances and degrees in each direction to the corresponding axe in this standard compass coordinate system. For example, the relative representation using laser range finding of "3.62m 35° South-East and 45° elevation", can be projected to a local CCRS (Figure 3).

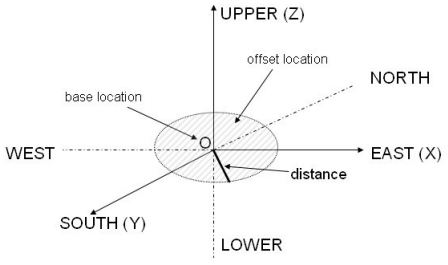

Fig. 2. A centered relative representation

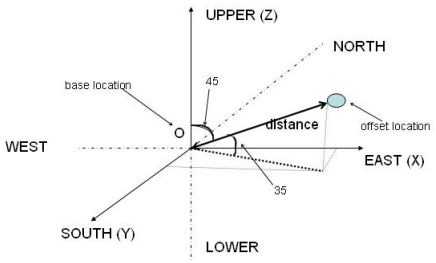

Fig. 3. A adjacent relative representation

3.2 Representation of Spatial Relationships

We have shown that different models (set-based, graph-based, and hierarchical models) have their particular strength on expressing spatial relationships. However, none of them

can work individually to express all the spatial relationships well. Our space model unifies a lattice and graph model. It describes *containment*, *disjointness*, and *overlapping* relations using its lattice model; and the *adjacency* and *connectedness* relations, and notions of *distance* using its graph model.

A Lattice Model Among hierarchical models, a lattice model has better expressivity and flexibility than a tree model. In the following, we will introduce a tractable way to organise symbolic spaces in a lattice model according to the containment relation.

Definition 1. *All the symbolic spaces in a real world are organised into a set S with the containment relation $\leq$.*

- *The space set $S = \{s_1, \ldots, s_n\}$;*
- *Containment relationship: $s_i \leq s_j$ means that a space s_i is contained by s_j. The containment relation is a partial order:*
  - *Reflexivity: $s \leq s$ means a space s contains itself;*
  - *Antisymmetry: $s_i \leq s_j \wedge s_j \leq s_i \Rightarrow s_i = s_j$;*
  - *Transitivity: $s_i \leq s_j \wedge s_j \leq s_k \Rightarrow s_i \leq s_k$.*

In Figure 4, the left side simulates part of the real world space, and the right side organises its symbolic spaces.

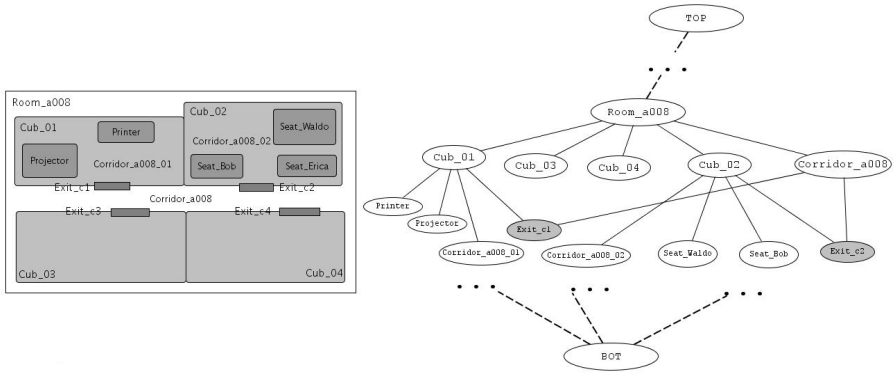

Fig. 4. An example of a lattice model for a real world space

A particular containment relationship $s_i \preceq s_j$ means that a space s_i is immediately contained by s_j. The *immediate containment* $\preceq$ is the basic operation to construct *containment* $\leq$. In building a spatial model, only the immediate containment is specified for spaces, while the general containment can be derived transitively. Their conversion relation is defined as follows:

- If $s_i \preceq s_j$ and whenever $s_i \leq s_k \leq s_j$, it follows that $s_k = s_i$ or $s_k = s_j$.

– $s_i \leq s_j$ holds iff there exists a finite sequence of spaces $s_{k1}, \ldots, s_{km}$ such that $s_i \preceq s_{k1} \ldots s_{km} \preceq s_j$.

According to both containment relationships, we derive (immediate) sub / super spaces. For any space $s \in S$,

– its immediate sub spaces: $s.isub = \{s_i \in S | s_i \preceq s\}$;
– its sub spaces: $s.sub = \{s_i \in S | s_i \leq s\} \supseteq s.isub$;
– its immediate super spaces: $s.isup = \{s_i \in S | s \preceq s_i\}$;
– its super spaces: $s.sup = \{s_i \in S | s \leq s_i\} \supseteq s.isup$.

After discussing the containment relation, we start analysing the way to construct the space set. That is, how the whole space is partitioned into spaces with varying levels of granularity step by step. In the space set S, there are two special spaces: one is the universal space $s_\top$ that represents the whole world; another is the null space $s_\bot$ that represents non-existing space. That is, $\forall s \in S$, $s \leq s_\top$ and $s_\bot \leq s$.

Proposition 1. *For a space $s \in S$, its immediate sub spaces are $s.isub = \{s_i \in S | s_i \preceq s\}$, satisfying*

– *the union of $s.isub$ covers the whole space represented by s.*
– *the intersection of any two of its immediate sub spaces s_i and s_j can be*
  - *$s_\bot$, if they are disjoint;*
  - *a unique space $s' \in S$, if they are overlapping. s' is the common space with the coarsest granularity, which follows that $s_k \leq s'$, where $s_k \in s_i.sub \cap s_j.sub$.*

Proposition 1 provides a tractable top-down approach to divide a space into finer-granularity spaces gradually. A space can be partitioned into a set of spaces in different orthogonal planes. For example, a building can be partitioned into floors in a vertical plane; and it also can be partitioned into wings in a horizontal plane. Both the vertical plane and the horizontal plane are orthogonal to each other. It's obvious that floors overlap with wings. An overlapped space is the unique space among the intersection of sub spaces of the corresponding floor and wing, e.g., "wing A at the ground floor".

In this lattice model, the overlapping relationship has a general meaning, which includes overlapping in the same plane and that in different orthogonal planes. The overlapping in the same plane can be slightly overlapping and partially overlapping. The slightly overlapping happens when a space is neighboring to another, while these spaces cannot be simply separated. There usually exists a common space that is hard to decide which space it belongs to. From figure 4, if a cubicle Cub_{01} is adjacent to a corridor $Corridor_{a008}$ and another cubicle Cub_{02}, there is an exit $Exit_{c1}$ and a wall between them respectively, which can not be simply decided which room the exit or wall should exactly belong to. However, there is not a delicate approach to determine whether overlapping is slight or partial. To simplify, both overlapping is regarded as the same, and it is only necessary to evaluate whether the overlapping spaces are passable or not. When two spaces are connected through a third space, this space is called a *sensitive space*.

Definition 2. *For a space $s \in S$, if any two of its immediate sub spaces s_i and s_j overlap in the same orthogonal plane, then a* ***sensitive space*** *is the unique space s_* that is the overlapping space with the coarsest granularity among the intersection of sub spaces of s_i and s_j.*

The semantics of a sensitive space is to traverse coarser-granularity spaces, labeled as $s_i \Leftrightarrow s_j$. Across a sensitive space, an object can pass through the coarser-granularity spaces. There are many types of sensitive spaces. For example, a hall traverses two wings, a lift traverses floors, and so on.

A poset (partially ordered set) can be a complete lattice if there exist the greatest lower bound (meet $\sqcap$) and the least upper bound (join $\sqcup$) for any subset A of S. For any two spaces, join is defined as the unique space with the finest granularity that covers both of them; meet is defined as the unique common space with the coarsest granularity that is covered by both. According to the above construction procedure, the partially-ordered spatial set S can be constructed into a lattice model. Proposition 1 makes sure the existence of the join and meet for any two spaces. The leftmost in Figure 5 is a real space, where s_i and s_j overlap at two finer spaces A and B, as the figure (a) shows. According to the proposition, only one overlapping space is allowed, so a common space s' is created to be an immediate super space of A and B (see Figure 5 (b)). In the

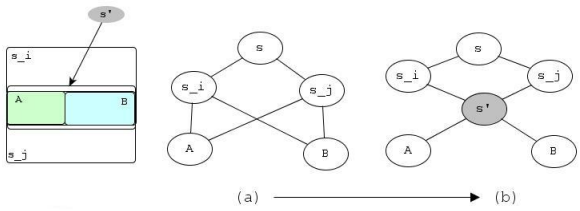

Fig. 5. A special case in partitioning

following, we give a formal definition for join and meet. Given two spaces s_i and s_j,

- *join operation*: $s_i \sqcup s_j = s$, where $\forall s' \in s_i.sup \cap s_j.sup$, $s' \leq s$;
- *meet operation*: $s_i \sqcap s_j = s$, where $\forall s' \in s_i.sub \cap s_j.sub$, $s \leq s'$.

Figure 6 shows typical relations between spaces in a lattice model. Some examples of the join and meet operations between these spaces are computed as follows. In Figure 6 (a), $s_i \sqcap s_j = s_i$, and $s_i \sqcup s_j = s_j$. In Figure 6 (b),

$$\begin{aligned} s_i \sqcap s_j &= c, & s_i \sqcup s_j &= s_k; \\ A \sqcap s_i &= s_\perp, & A \sqcap B &= s_\perp; \\ A \sqcup s_i &= s_k, & A \sqcup B &= s_k. \end{aligned}$$

In the above, a lattice model is constructed for symbolic spaces according to the containment of architectural design, which portrays the physical view of an environment.

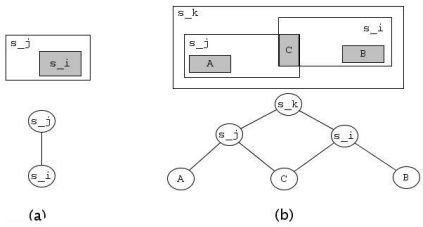

Fig. 6. Simple examples of spatial relationships in a lattice model

For the same environment there may be multiple lattice models with respect to different partition principles. For example, another lattice model can also be built in terms of the administrative functions, such as lab area, lecture area, or office area, which portrays a conceptual view of an environment. Both lattice models present various views of an environment, which can be beneficial in serving customised queries. Take an example of a query: "a reception office in this building". It is more efficient to execute this query in the latter lattice model that will only look for the reception office in the administration area, rather than search the whole space in the former lattice model.

So far, this section has described how to organise the symbolic spaces into a partial order set according to the spatial containment relationship. The proposition 1 provides a top-down approach to dividing the whole space $s_\top$ with the coarsest granularity into spaces with finer granularities. It also helps to constrain the constructed poset S to be a lattice model that is bound by the universal space $s_\top$ and the null space $s_\bot$.

A Graph Model The above lattice model is used to explore the spatial containment, overlapping, and disjointness relationships, while the following graph model will help to make the adjacency and connectedness relationships explicitly.

Faced with the huge complexity of spaces in reality, it is difficult to build a large graph for the entire space and maintain the integrity of the graph. Some adjacent spaces may be ignored or missing when drawing the adjacency relation for a certain space. The large graph needs lots of effort to maintain when a space is changed (such as enlarged, detracted, or deleted). Instead, our approach collapses a large graph into smaller graphs, each of which represents a separate region. These small local graphs are associated with the immediate sub spaces of each node in the lattice model. Therefore, when needed, these smaller graphs can be composed together as well.

Definition 3. *For any $s \in S$, a corresponding **graph** $G_s = (V, E, E_c)$ is a directed and weighted graph, where*

1. *V is the immediate sub spaces of s; that is, $V = s.isub$, where $s \neq s_\bot$ and $\forall s' \in s.isub$, $s' \neq s_\bot$;*
2. *E represents the adjacency relation. s_i — s_j means s_i and s_j is neighboring to each other;*

3. *E_c represents the connectedness relation. $s_i \rightarrow s_j$ means that an object can go from s_i to s_j;*
4. *d on each edge represents a distance between spaces.*

If spaces are disjoint to each other or overlapping in different orthogonal planes, there are no edges between them; otherwise, they are considered *adjacent* to each other. Further, if a space is passable to another directly, the relationship between them is regarded as *connectedness*. Particularly, the connectedness relation implies direction. When an object can go from a space s_i to s_j, then s_i is connected to s_j, labeled as $s_i \rightarrow s_j$. In some circumstances, it is forbidden to go from s_j to s_i (e.g., an escalator).

Spatial Relationship	Evaluation
containment	$s_i \sqcap s_j = s_i$ (or s_j) $s_i \sqcup s_j = s_j$ (or s_i)
disjointness	$s_i \sqcap s_j = s_\perp$
overlapping	$s_i \sqcap s_j \neq s_\perp$
adjacency	$s_i \sqcap s_j \neq s_\perp$ and s_i — s_j
connectedness	$s_i \sqcap s_j \neq s_\perp$ and $s_i \rightarrow s_j$

Table 1. Evaluation of different spatial relationships

A local graph for a node in a lattice model reflects a certain abstraction level of spaces, which is a certain scale of observing the real space. For example, Figure 7 presents two graphs with varying granularities, which correspond to the spaces in Figure 4. The graph mapping to $room_{a008}$ reflects the adjacency of cubicles, while the graph mapping to Cub_{01} reflects the adjacency of finer spaces within a cubicle. It is convenient for applications to flexibly load graphs in different scopes. For example, when an object is in the ground floor, the application only needs to present the graph of rooms that are located in the ground floor; when an object moves into the room $room_{a008}$, it presents a more detailed graph of cubicles; furthermore, when the object enters a cubicle, a graph with finer granularity is presented.

This approach decreases the complexity of loading a huge graph in the beginning, and it reduces the amount of information because it does not need to load unnecessary graphs such as Cub_{02} when only the graph Cub_{01} is needed. This approach makes it advantageous to maintain a relatively small graph, because a graph is part of the whole world. It also makes less effort to reconfigure the graph without affecting other graphs.

This approach also keeps the spaces at the same abstraction level in the same graph. It makes more sense when we discuss the adjacent relation between the spaces at the same level of abstraction (such as among rooms or among buildings). However, it is hard to discuss the adjacency between a room and a floor.

Distance is a complicated and application-specific notion. A distance is a physical length between two spaces; however, sometimes it is related to other contextual information (such as transportation, or path restriction) in real applications. However, this graph model will only consider a general definition for *physical distance*, while

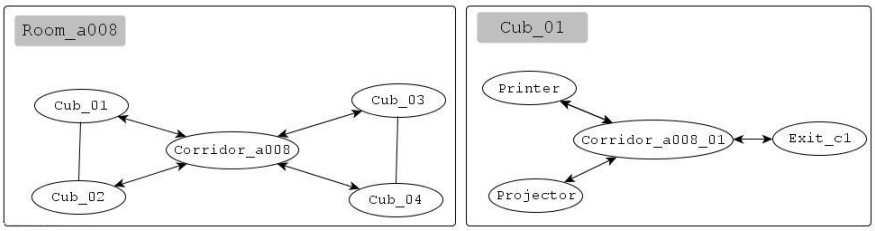

Fig. 7. An example of two graphs at different abstraction levels

other contextual information on distance can be extended in customised applications. A physical distance can be classified into *absolute* and *accessible* distances. An absolute distance is a direct length between two spaces, while an accessible minimal distance is a length along a path through which a space is connected to another space. An accessible minimal distance is related to path finding, and it can be computed by accumulating a series of absolute distance. Therefore, absolute distances are fundamental for the distance concept.

In the graph model, an absolute distance d on each edge is a tuple $< d_e, d_l >$, which corresponds to the Euclidean and path distance respectively. The Euclidean distance is the shortest straight line length between centers of two spaces, while the path distance is the length of a path from the center of one space to the center of another space. This approach to describe distance is borrowed from MiddleWhere [19].

Path finding is an important issue in a location-aware system. That is, how a path is located from a source space to a target space. There exists a set of paths between two given spaces.

Definition 4. *A **path** consists of some finite space sequence following the connectedness relation, which starts from a source space s and ends at a target space t.* $p(s,t) = s \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow t$.

Local graphs cannot satisfy the requirements of path finders if the given source and target spaces are not in the same local graph. Thus a larger graph is required, making it necessary to combine a set of local graphs. If all the spaces in the graphs have geometric representations, then these spaces can be projected into a uniform Cartesian coordinate reference system (CCRS). Thus a combined graph is produced by computing their coordinates. If not all of the spaces have geometric representations, we propose an alternative approach to integrate graphs through sensitive spaces. In a lattice model, coarser-granularity spaces are traversed by their sensitive space. If two coarser spaces s_i and s_j have a sensitive space s_* between them, when two graphs G_{s_i} and G_{s_j} are required to be merged, s_* will be located and serve as a connection of two graphs. Section 4 will give a description of the path finding algorithm.

4 Demonstration

To demonstrate the applicability of our space model, it is applied to build a map for our computer science building. All the spaces are represented in a resolute hybrid representation. They are organised in a lattice model, and then a graph model is built for each node in a lattice. In the following, we will describe how spaces are expressed in hybrid and relative representations. Later, we will detail how the space model helps in path finding algorithms.

4.1 Spatial Representation

Each space has a hybrid representation. A symbolic representation assigns a human-understandable name for the space, while a geometric representation characterises the space's geometric shape and a list of coordinates under a certain CCRS. For example in Figure 4, a space $room_{a008}$ is represented in Figure 8. The space object deals with all location-related information, which is a property of an entity. The entity lab_{XRS} carries with other contextual information like its included persons, calendar, and research areas.

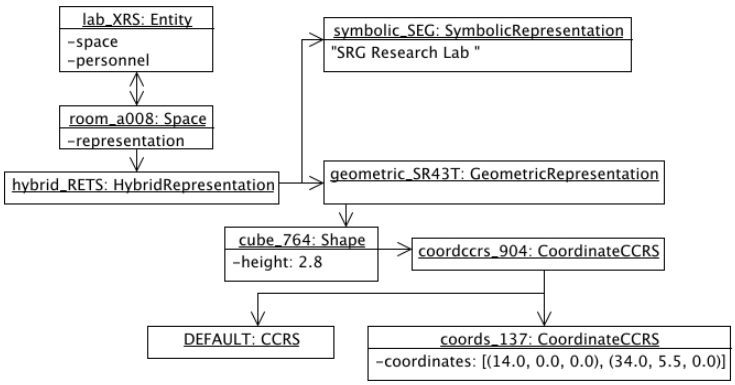

Fig. 8. A typical example of representing an entity's space

Section 3.1 has provided a general way to express a relative representation; however, its typical applications are simple, like "a place 500m east of this building". This is represented as follows, and a resolute location can be computed in applications by building a local CCRS with the base as an origin point and the standard compass direction as three axes' direction. The offset representation can be projected into this CCRS according to the distance to the base representation, and degree to each direction axis.

Base Representation	Offset Representation		
	distance	direction	degree
$building\_space\_SRES$	5.0	EAST	0.0

4.2 Path Finding

In this section, we will describe how the lattice model integrated with graphs helps to improve the efficiency of a typical path finding algorithm, while a concrete path finding algorithm is out of the scope of this paper.

Our space model is used to partition the world into small sub graphs that are organised and managed under the lattice model. Different levels of the lattice model determine the abstraction levels of each sub graph. At a certain abstraction level, each sub graph is further partitioned into smaller local graphs that are connected to each other through sensitive spaces. We propose that the path finding algorithm will be executed on each local graph, rather than throughout a huge graph for the whole environment.

Given two spaces s_{source} and s_{target}, their immediate super spaces are located from the lattice model, labelled as G_s and G_t. If G_s and G_t are the the same graph, the searching will be only executed on the spaces in that graph, while ignoring all the other spaces and spatial relationships out of that graph. If G_s and G_t are different graphs, a series of sensitive spaces, called *sensitive path*, will be located as follows: $G_s \xrightarrow{s_1} G_1 \quad \cdots \quad G_{m-1} \xrightarrow{s_m} G_t$, where $s_1, ..., s_m$ are sensitive spaces, and $G_1, \ldots, G_{m-1}$ are the graphs connecting G_s and G_t. This shows the higher level hints of searching: sensitive spaces are the key spaces between the given source and target spaces. In each local graphs, the paths will be searched, such as $s_{source} \rightarrow s_1$, $\ldots$, $s_i \rightarrow s_{i+1}, \ldots, s_m \rightarrow s_{target}$.

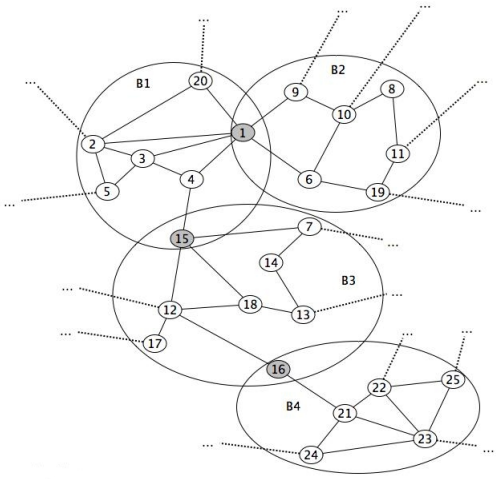

Fig. 9. A simple example of path finding

For example in Figure 9, a path from a space s_2 to s_{23} is required. The smallest local graphs for s_2 and s_{23} are located as B_1 and B_4, both of which are not directly connected

to each other. By examining the sensitive space set, a series of sensitive spaces sensitive spaces and the intermediate graphs will be determined: $B_1 \xrightarrow{s_{15}} B_3 \xrightarrow{s_{16}} B_4$. Following this, paths will be searched only in each local graph , such as $s_2 \rightarrow s_{15}$ in B_1, $s_{15} \rightarrow s_{16}$ in B_3, and $s_{16} \rightarrow s_{23}$ in B_4.

It is worth noting that a given source or target space may be contained in more than one smallest local graphs, if it is a sensitive space. In the above figure, the sensitive space s_1 has two immediately super spaces B_1 and B_2. Besides, there are multiple possibilities of sensitive paths. Assume that the space B_2 is directly connected to B_3 through the sensitive space s_7. For the above example, another series of sensitive spaces will be produced: $B_1 \xrightarrow{s_1} B_2 \xrightarrow{s_7} B_3 \xrightarrow{s_{16}} B_4$. Correspondingly, another path finding will be carried out: $s_2 \rightarrow s_1$ in B_1, $s_1 \rightarrow s_7$ in B_2, $s_7 \rightarrow s_{16}$ in B_3, and $s_{16} \rightarrow s_{23}$ in B_4. As the path finding is executed separately on local graphs, the local results can be shared between different series of sensitive paths. For example, the path $s_{16} \rightarrow s_{23}$ in B_4 can be shared from the above example.

The path finding procedure may be complicated by the above factors such as multiple immediate super spaces and multiple possibilities of series of sensitive spaces. However, the number of immediate super spaces is usually small, and the searching of sensitive paths is the procedure of finding a path between the immediate super spaces. The computation cost is relatively small, compared to searching a huge graph covering the whole space. This space model not only reduces the searching space for possible paths, but it also provides a higher level view of searching by locating a series of sensitive spaces.

Besides this path finding algorithm, the model also can answer nearest neighbor queries (e.g. finding the nearest printer [13]) by considering the distance on each edge. It is difficult for hierarchical location models to carry out path finding, but it is possible for them to answer the nearest neighbor query. This query is carried out by searching all the printers, computing the distances between the user and all the candidate printers and choosing the closest printer. This approach ignores the potential path inherent during searching. It is obvious that the shortest absolute physical distance between two coordinate points is not the shortest accessible distance along the available path. Besides, the simple computation of distances will not help a stranger who is not familiar with that space. However, our model can find the nearest neighbor through choosing the shortest accessible distance along a path.

Compared to other graph models, this space model can reduce the searching space largely during a path finding. A typical graph model would carry out searching a huge graph that consists of all the spaces in the entire environment. In the above example, the searching probably starts from all the spaces connected to s_2 in the whole graph. However, the path finding algorithm based on our space model will always be executed on the limited number of spaces in local graphs, and connected spaces in other local graphs are transparent. Only when the searching fails, it will be extended to other local graphs through sensitive spaces. The searching space is only the number of spaces in potentially local graphs and sensitive spaces, which is relatively small compared to searching all the spaces in the whole graph at once.

5 Conclusion and Future Work

This paper has proposed a unified space model to service the requirement of spatial representation and relationships for a large-scaled and complex environment. This model provides a powerful and expressive representations of location information for any space. It supports traditional symbolic and geometric representations, and relative location representations, which is more comprehensive than existing models.

This space model takes the advantages of both lattice theory and graph theory, and combines them in a novel way. This single model supports comprehensively spatial relationships such as containment, adjacency, and connectedness. It offers multiple graphs at various abstraction levels that are distributed at different nodes in a lattice model. The demonstrations in Section 4 suggest that this will potentially improve the efficiency of path finding algorithms.

This space model is comprehensive, for it involves all the details that are needed to represent location-related information. It can be simplified to suit to simpler applications. When graph models are not needed, developers can only construct a lattice model without building the connectedness relationship. When a hierarchical model is not needed, developers can just build a typical graph model while adding a universal space to be the immediately super spaces for all the spaces in this graph. Considering the natural complexity of real environments, it always takes a huge effort to build a detailed space model, no matter based on our model or existing models. However, our model tries to reduce the complexity of construction by collapsing a whole graph into manageable local graphs, and linking and organising them into a lattice model. Moreover, this approach can reduce the cost of maintenance. When spaces in the environment have been updated (e.g., adding or removing spaces), it will be easier to track down a few of local graphs that are required to be revised correspondingly, rather than check and affect the whole graph. However, if the spaces being updated are (or will be) sensitive spaces, it may cause inconsistency. This will increase the complexity of maintenance in our model.

To make better use of this model, developers must have a good knowledge of the target environment, including partition principles, levels of granularity, and particularly the sensitive spaces being chosen. Developers should build a lattice model for all the spaces in an environment with the help of Proposition 1 to locate sensitive spaces. Then they can build connectedness relationships between immediately sub spaces for each node in the lattice so as to form local graphs. This ensures the consistency of spatial relationships between all the spaces. In our initially simple experiment, this space model can be easily constructed for the spaces in one floor. However, to attain the goals of the model, it should be applied and evaluated in a larger-scaled and more complicated environment. In parallel, the path finding algorithm must be properly evaluated and refined as the space set grows.

Furthermore, the space model could be extended with extensions that constrain the accessability of locations, e.g. opening hours for public spaces, highway restrictions for roads, or gender restrictions for bathrooms [5]. Further constraints could be set to determine security and privacy characteristics of location information.

References

1. M. Bauer, C. Becker, and K. Rothermel. Location models from the perspective of context-aware applications and mobile ad hoc networks. *Personal Ubiquitous Computing*, 6(5-6):322–328, 2002.
2. C. Becker and F. Durr. On location models for ubiquitous computing. *Personal Ubiquitous Computing*, 9(1):20–31, 2005.
3. M. Beigl, T. Zimmer, and C. Decker. A location model for communicating and processing of context. *Personal Ubiquitous Computing*, 6(5-6):341–357, 2002.
4. B. Brumitt and S. Shafer. Topological world modeling using semantic spaces. In *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing*, Atlanta, Georgia, September 2001.
5. H. Chen, T. Finin, and A. Joshi. An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197–207, May 2004.
6. P. Coschurba, K. Rothermel, and F. Durr. A fine-grained addressing concept for geocast. In *ARCS '02: Proceedings of the International Conference on Architecture of Computing Systems*, pages 101–113, London, UK, 2002. Springer-Verlag.
7. S. Domnitcheva. Location modeling: State of the art and challenges. In *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing (Ubicomp 2001)*, Atlanta, Georgia, September 2001.
8. F. Durr and K. Rothermel. On a location model for fine-grained geocast. In *Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*, pages 18–35, 2003.
9. R. Glassey and R. I. Ferguson. Location modeling for pervasive environments. In *Proceedings of the 1st UK-UbiNet Workshop*, September 2003.
10. D. Graumann, J. Hightower, W. Lara, and G. Borriello. Real-world implementation of the location stack: The universal location framework. *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2003)*, 00:122, 2003.
11. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.
12. H. Hu and D. L. Lee. Semantic location modeling for location navigation in mobile environment. In *5th IEEE International Conference on Mobile Data Management (MDM 2004)*, pages 52–61, Berkeley, CA, USA, January 2004.
13. C. Jiang and P. Steenkiste. A hybrid location model with a computable location identifier for ubiquitous computing. In *UbiComp '02: Proceedings of the 4th International Conference on Ubiquitous Computing*, pages 246–263, London, UK, 2002. Springer-Verlag.
14. W. Kainz, M. J. Egenhofer, and I. Greasley. Modeling spatial relations and operations with partially ordered sets. *Geographical Information Systems*, 7(3):215–229, 1993.
15. M. Korkea-aho and H. Tang. A common data set and framework for representing spatial location information in the internet. *Cluster Computing*, 5(4):389–397, 2002.
16. N. Marmasse and C. Schmandt. A user-centered location model. *Personal Ubiquitous Computing*, 6(5-6):318–321, 2002.
17. C. A. Patterson, R. R. Muntz, and C. M. Pancake. Challenges in location-aware computing. *IEEE Pervasive Computing*, 2(2):80–89, 2003.
18. S. Pradhan. Semantic location. *Personal Ubiquitous Computing*, 4(4):213–216, 2000.
19. A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas. Middlewhere: a middleware for location awareness in ubiquitous computing applications. In *Proceedings of Middleware '04*, pages 397–416, New York, NY, USA, 2004.
20. A. Schmidt, M. Beigl, and H.-W. Gellersen. A location model for communicating and processing of context. *Personal Ubiquitous Computing*, 6(5-6):341–357, 2002.