This chapter is about the design of spatial representations.

5

Reasoning about Space

When people discuss the breadth of human reasoning abilities, they usually include spatial reasoning. Spatial reasoning is ubiquitous in daily life. People remember where they keep things and put them away. They navigate from place to place. They fit jigsaw puzzles together. They design mechanical devices like engines, whose operation depends on the arrangement and motion of their deliberately shaped parts. Reasoning about space is central to all these activities. Within the context of computer systems, algorithms for machine vision, solid modeling, and robot planning are dominated by concepts of spatial and geometric reasoning.

Does spatial reasoning require specialized representations? In some cases, spatial reasoning can be carried out without a comprehensive model of space. In the folk tale *Hansel and Gretel*, two children do not know their way in the woods. They find their way home by retracing a trail of bread crumbs that they left behind. The environment records the memory of the trail for them; the children perceive the memory as they see and follow the crumbs back home. In this example, much of the reasoning work that might be done with an internal spatial representation is instead off-loaded to objects in the environment and the children's perception of those objects.

Not all spatial reasoning tasks lend themselves to relying on the environment as primary spatial memory. Controlled experiments, both with animals and people, suggest that internal spatial representations are used and perhaps necessary to do many common tasks efficiently, such as rats finding paths in mazes (Tolmen, 1948) or people identifying identical parts from different perspective views (Balsam, 1988).

So what do spatial representations represent? In the preface to his book *Computation and Cognition* (1984), Zenon Pylyshyn wrote:

> We conceive of space as a completely empty, infinite, three-dimensional, isotropic, disembodied receptacle distinct from the earth or any object that might be located on

> the earth. . . . Such a strange idea was unthinkable before the seventeenth century; . . . not until Newton was the task of "geometrization of the world" completed.

This "geometrization" is the invention of what is sometimes called Euclidean space or the Cartesian coordinate system. It is the familiar characterization of space in terms of x, y, and z coordinates. Given that a simple coordinate system can represent space, what else is needed?

The representation requirements for spatial reasoning—like the requirements for any kind of reasoning—vary greatly with the particulars of a task. Reasoning about space can involve reasoning about volumes, surfaces, or distances. It can involve reasoning about shapes and intersection. Exactly what is involved is as open-ended as the tasks at hand. Choice of representation is important. Particular representations can make certain computations much more efficient than others. Different representations have different expressiveness. They have different efficiencies for particular kinds of reasoning and scale differently with the number of relationships being encoded.

In the following we consider a sequence of cases involving spatial reasoning. We have arranged the cases to provide a tour of considerations and approaches in supporting spatial reasoning. Our goal is not to provide a prescriptive set of representations. Rather, it is to illustrate fundamental choices one makes in designing spatial representations for knowledge systems.

5.1 Spatial Concepts

Euclidean space contributes several concepts that we employ in our discussion of spatial reasoning. We list some of these now, to be explicit about their role in our representations.

Points are atomic locations in space. Depending on the task, the space itself can be represented as being either two-dimensional (2-D) or three-dimensional (3-D). Points are referenced by their coordinates. As with temporal representations, the choice between real and integer coordinates depends on the application. The space as a whole is a set of points. Lengths are one-dimensional measures of distance in space, areas are two-dimensional, and volumes are three-dimensional. A **direction** can be defined as a point on the unit sphere, or a vector from the center of the sphere to some point on its surface. A **region** is a set of points. Two-dimensional rectangular regions, the set of points included in the edges and interior of a planar rectangle, are an important special case.

These spatial concepts are used in maps. For example, zoning regions for city planning are usually shown as superimposed over street maps. Office assignments and phone numbers in an office building are often shown as superimposed on two-dimensional maps of the floors of a building. Route planning is often illustrated in terms of selections over a map of highways and streets. Historically, maps have been sold as paper-based information products. Increasingly, map data are commercially available on computers. Not only are world and region maps available for computer-based browsing, but geographically organized databases also are widely available with which other data can be integrated.

Spatially indexed databases are databases in which location data are used to retrieve information about objects in the database. Such databases are analogous to the temporal databases discussed already, in which time information is used to retrieve information about events. In spatial databases, coordinate information is used to retrieve information about region properties or nearby objects. Spatial databases support spatially oriented queries characteristic of

spatial problem solving and typically provide a visual user interface that displays images of the objects in their locations on a map.

In the production and interpretation of maps, certain conventions are used that determine what a map communicates to its users. For example, map data are supposed to be complete relative to the information types being displayed: A street map of a city is supposed to show all the streets. A room map of a building is supposed to show all the rooms. Such conventions support spatial versions of the closed-world assumption (CWA).

The nature and specification of these assumptions is especially crucial when maps are partial, dynamic, and under construction. Consider the construction of maps when people explore new territory or the dynamic world models created in systems for robot perception. In both cases, the maps represent the best-known interpretations of a scene at a given time. These dynamic cases require that the spatial representation include a means for characterizing ambiguity and incompleteness of information.

Here are some representational distinctions related to the closed-world assumptions that arise in maps:

1. *Data completeness:* Can it be assumed that the map shows every object that is in the real-world region represented by the map?

For example, in a map of new wilderness, if no potable streams are shown, does that mean there are none? Does it mean that there are no streams, or that streams are seasonal, or that their potability is in question? When a map is known to be incomplete, it can be useful to have a means to signify that some regions are known not to have certain kinds of objects. Without such notation, there is no way to distinguish the *known absence* of objects from the *absence of knowing* about any objects. Sometimes it is useful to indicate which regions have been explored and which have not. If the map is pieced together from parts by different explorers, then information about which explorers visited each region may be useful in determining what kinds of information may be complete or not in the map.

2. *Object uniqueness:* When can we assume that two separate object descriptions in the map correspond to two different real-world objects?

Suppose two explorers both report seeing ruins, but at somewhat different though close coordinates. Under what circumstances might we decide that they saw the same ruins?

3. *Object piecework:* If only separate parts of what might be one object have been perceived, how can their representations in the map be identified as being possibly (but not definitely) the same object?

Consider a situation where some mountains in one region have been explored by one group and mountains in another region have been explored by another group. Do the mountains make up a single range?

4. *Multiple levels of resolution:* Must all parts of a map present information to the same detail?

Travel maps are sometimes divided into sections with different levels of detail. A travel map for the San Francisco Bay area shows major highways for both sides of the bay and also

detailed street maps for San Francisco, San Jose, and Oakland. The highway maps support planning routes between cities and the street maps support it within cities. Not only would a detailed street map of the entire Bay Area be unwieldy, but it also would present more complexity than is needed for planning highway travel.

The use of map sections with different levels of detail supports **hierarchical route planning**. In computer databases, multiple-resolution spatial representations are intended to serve similar needs. A **multiple-resolution spatial representation** is any representational technique that provides spatial information at more than one level of detail. Multiple-resolution representations can reduce the complexity of information that must be processed at the more abstract stages of reasoning, where too much spatial detail would be an unnecessary burden.

5.2 *Spatial Search*

Perhaps the simplest way to represent spatial data is to incorporate them into a relational database along with other data. Consider the following relational table of object names, colors, and coordinates.

Object	*Color*	*Location*	*Width*	*Length*
A	Red	(1, 5)	4	5
B	Blue	(3, 6)	3	10
C	Green	(2, 4)	6	9
. . .				

If the search to answer a query proceeds sequentially through the list, then the expected time is proportional to the length of the list. In a general-purpose relational database, spatial reasoning has no privileged position. Questions such as "Where is object A?" are answered in the same way as "What color is object B?" Questions such as "Does object A intersect object B?" require reasoning abilities beyond retrieval from the table. Relational and other general-purpose databases provide no particular support for spatial reasoning.

In contrast, suppose you are given a highway map of California and are asked to find the nearest city of more than 15,000 people near Santa Barbara. You would not start searching with Eureka, which is hundreds of miles away. You would probably first locate Santa Barbara on the map and then search through widening circles around its location until reaching a city of the required size.

In this way, the map supports reasoning about distance. Cities that are near each other in California are represented by dots that are near each other on the map. By searching in widening circles, we exploit the structure of the map to guide the search for a city. Computer searches through spatial databases that behave similarly are called spatial searches. A **nearest-first search** is one that considers candidate objects in order of increasing distance from a given object. Nearest-first search finds the closest object to the reference object that satisfies the search criteria. Some search methods approximate the nearest-first order of search.

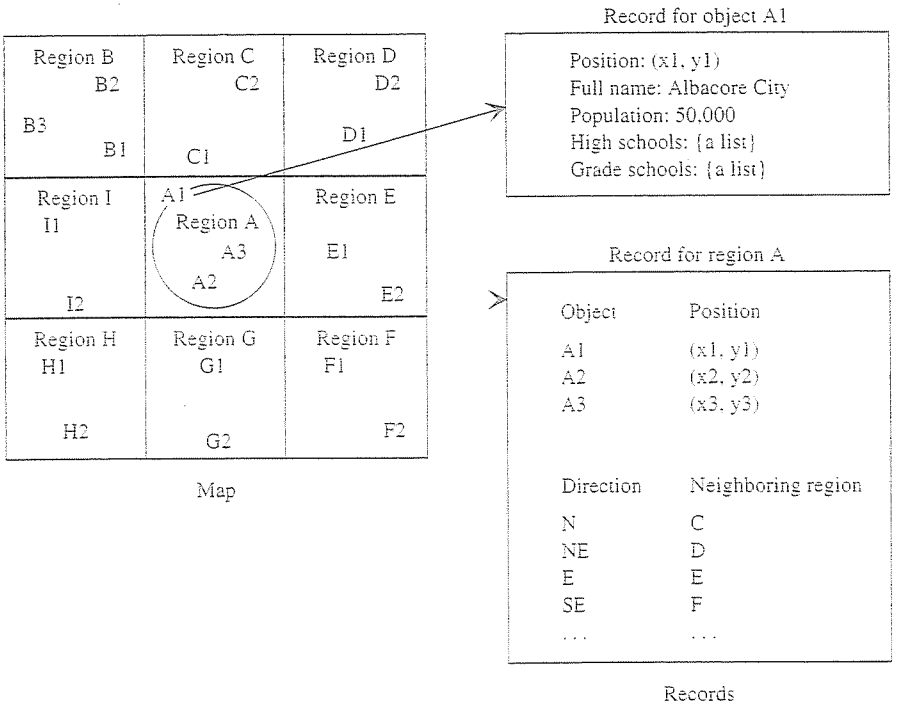

FIGURE 5.1. A representation that divides space into uniform rectangular regions.

5.2.1 *Simple Nearest-First Search*

The following example illustrates one way to implement nearest-first search in a computer database. Figure 5.1 shows an area divided into equal-size, nonoverlapping regions shaped as Manhattan rectangles. Manhattan rectangles offer economies for representing their boundaries and for computing whether a given point is inside or outside a region. In Figure 5.1 regions are named by letters such as A, B, and C. Objects within the regions are given numbers. Thus A3 indicates an object in region A. The coordinates for each object correspond to its position in the figure.

Each region is represented by a record that gives its location, pointers to the records representing neighboring regions, and a list of all the objects contained within the region. One advantage of uniform-size Manhattan regions is that the region records can be kept in an array such that the index of a region record in the array can be computed easily from the region coordinates (see exercise 2). Each object in the region is represented by a record giving its attributes. To provide an efficient means for locating an object's record given the object's name, the names can be organized in records in a balanced tree, a hash table, or another data structure with appropriately low access times.

One way to approximate nearest-first search is to order regions to be searched according to their nearness to a point. Within each region, search through objects is exhaustive in any sequential order. There are two basic policies to be determined:

- What region should be searched next?
- When can searching be terminated?

To motivate the design of a search policy, let us first look at an example. Consider a nearest-first search out from the reference point (or "target") A1 in Figure 5.1. The closest object to A1 is C1 in the neighboring region. A1 is in the upper left quadrant of region A. Intuitively, it makes sense to search region B before searching region D. One way to show this is to observe that every point in the lower right quadrant of region B is closer to A1 (or other points in the upper left quadrant of A) than is *any* point in region D.

This example is suggestive of policies for simple, exhaustive, nearest-first search based on rectangular regions. The next region to be searched should be an unsearched region containing the closest points to the reference point. If no object has yet been found that satisfies the search criteria, then search does not stop until the database is exhausted. If a satisfying object has been found, search terminates when all the unsearched regions contain only points farther away from the reference point than the closest found object.

Figure 5.2 gives pseudocode for a method for simple nearest-first search, SNFS-1. This method is given a reference object and some criteria. It returns the closest object to the reference that satisfies the criteria. If more than one object at the same distance satisfies the criteria, SNFS-1 returns the first object it finds.

The method is similar to best-first search and related methods from Chapter 2. A key to its operation is the function Get.Unvisited.Region, which returns the closest region to the reference (if any) that contains any points closer to the reference than the closest object found so far. In practice, tables can be precomputed that list the nearest neighbor regions given coordinates of the reference point in the current region. For example, given that the reference point is in the upper left coordinate of a region, neighboring regions in the directions W, NW, and N could potentially contain the closest objects.

5.2.2 *Problems with Uniform-Size Regions*

The uniform-region representation in the preceding section divides the total space into nonoverlapping regions that can be searched separately. Spatial search is approximated down to the level of regions. To prevent the exhaustive search of each region from requiring excessive time, there must be enough regions to divide up the objects into small sets.

When an area is divided into uniform-size regions, nearest-first search forces a space/time complexity trade-off. If the objects of interest are distributed nonuniformly, then some regions may have many objects and others will have none at all. This leads to potentially wide variations in retrieval performance. On the other hand, if the entire space is uniformly subdivided to the greatest resolution needed to divide any region, then the representation of a clustered but sparse data set may result in allocating many empty regions. When objects are clustered or otherwise

```
1.    Method Simple.Nearest.First.Search (reference, Criteria)

      /* Initialize */
2.    closest.object := nil; object.distance := VeryBigNumber;
3.    current.region := Region-of(reference)

4.    While current.region do
5.    begin

      /* Search through the objects in the region. */
6.          For object in current.region do
7.          begin
8.              If Criteria(object) and
                     Distance(object, reference) < object.distance
9.              then begin
10.                      closest.object := object
11.                      object.distance := Distance(object, reference);
12.                 end
13.   end

14.   Mark.as.Visited(current.region)

      /* Look for another region containing points
      as close as the closest object.*/
15.   current.region:= Get.Unvisited.Region(reference, object.distance)
16.   end

17.   Return(closest.object)
```

FIGURE 5.2. Method for simple nearest-first search (SNFS-1).

deviate from a uniform distribution, there is a choice between allocating records to represent empty regions and having some regions with too many objects in them.

Another problem with uniform regions arises during updating. If the database is changed dynamically, then adding a few new objects to a region can increase its object population beyond the performance-preserving limit. With uniform regions, the choice again is between having a region with too many objects and reassigning the region boundaries across the entire area.

5.2.3 *Quadtree Nearest-First Search*

Quadtrees provide a useful alternative for indexing spatial information, accommodating uneven object density and reducing the cost of updating. Quadtrees are regular, recursive decompositions of space into equal-size regions, each of which can be potentially further divided into

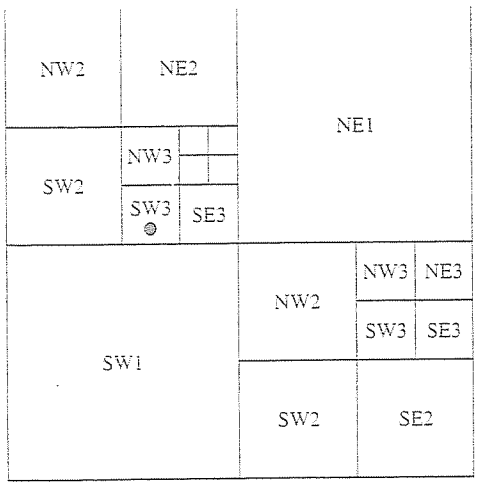

FIGURE 5.3. A hierarchical quadtree representation that divides space recursively into quadrants. Subquadrants are created as needed, to keep the number of objects per region within specified bounds.

equal-size subregions down to some greatest resolution. In the standard 2-D approach, all the subregions are one-fourth of the area of their parent region and are referred to as **quadrants**. To avoid a proliferation of empty quadrants, the division is done only as needed. When the number of objects in a quadrant exceeds a threshold, then the quadrant is subdivided further into subquadrants through as many levels as needed to limit the number of objects per region. An example of a quadtree is given in Figure 5.3.

The labels of the quadrants in Figure 5.3 indicate the level of the quadrant and its position in the decomposition of its parent. We use this label notation to explain navigation operations on quadtrees. The four parts of a quadrant are labeled NW (northwest), SE, SW, and NW, corresponding to the directions of the compass. The number indicates the depth in the tree.

Sequences of these quadrant labels are used as addresses of nodes in a quadtree, with the most significant parts on the left. For example, NW1.SE2.SW3 corresponds to the subquadrant in Figure 5.3 with the black-filled circle in it. Although the level numbers are redundant in these strings, we retain them here for clarity. Navigation operations on these addresses are needed for traversing or browsing the quadtree. Here are examples of navigation in Figure 5.3:

East (NW1.SE2.SW3) = NW1.SE2.SE3
West (SE1.SE2) = SE1.SW2
West (NW1.SE2.SW3) = NW1.SW2.SE3
North (SE1.NW2) = NE1.SW2
East (SW1) = SE1

Addresses in quadtrees have variable length because quadtrees have variable depth. The interpretation of these addresses in navigation requires handling exceptional cases that do not arise in fixed-length addressing schemes. For example, the length of the addresses returned by the navigation operations is always the same as the length of the given addresses. If the address is

too long, meaning that the addressed quadrant has not yet been instantiated to that level, this would be detected while following the address by noticing that the parent has no successors. In that case, retrieval or other reasoning can take place using the larger parent quadrant. If the address is too short, meaning that the addressed quadrant has successors but the address does not specify one of them, then there are other choices depending on the needs of the method. For example, it may be appropriate to search all the successors. (See the exercises for methods for navigation and nearest-first search in quadtrees.)

In summary, quadtrees improve the space/time trade-offs of representing regions with sparse and clustered objects. Although we have discussed quadtrees here only for the case of representing two-dimensional spaces, the generalization to three dimensions is straightforward. In addition, it is possible to divide regions into numbers of subregions other than four, such as nine.

5.2.4 *Multi-Level Space Representations*

Maps sometimes portray different parts of a region in different amounts of detail. To plan a trip from a home in Palo Alto to a restaurant in San Francisco, a visitor might use a street map of Palo Alto to plan a route to the highway, a bay-area highway map to plan the middle part of the trip, and a street map of San Francisco to plan a route from the highway to the restaurant.

This use of different amounts of map detail for different levels in problem solving is a variant of hierarchical search as discussed in Chapter 2, but specialized to spatial representations. In this section we focus on abstractions related to space, such as approximations of location and distance. Figure 5.4 presents some fragments from a database, which is organized in multiple levels of detail. In this example we have a detailed database of the roads, buildings, and rooms in a section of a large city. The database is intended to help people find their way around in the city and may be implemented in portable computers or "personal digital assistants."

The top level of our hierarchical representation is a city street map. A portion of that map is shown in Figure 5.4. The city-street level divides the city area into a grid of regions. Object locations are represented only in terms of the regions in which they are contained. Thus, The New Curiosity Shoppe is in the lower left region. Shapes are not explicitly represented, except insofar as that bounding boxes can be inferred since they are listed in all the regions in which they extend.

The key point about this level is that it shows only the "main objects of interest" for the problems at hand. In our example, the map shows key buildings, key landmarks (such as Murky Lake), and main thoroughfares. Minor streets are not included. The conventions we have chosen in this example are similar to those in many city maps in that we do not show all the buildings, driveways, and so on. The amount of detail is presumably appropriate for the purpose at hand; to guide driving.

The intermediate level of our database, the block map, introduces more detail. The example in Figure 5.4 portrays the information in the block level for the shaded region from the city-street level. This level shows not only the main thoroughfares, but also the side streets, parking lots, and walking paths. The purpose of this level is to provide information to guide parking and walking to the building. All buildings are shown, including those of less immediate interest such as Convenience Pharmacy. Positions and shapes are also shown in more detail. Boundaries of objects are described in terms of bounding polygons. In addition, the data about buildings include height in terms of number of floors.

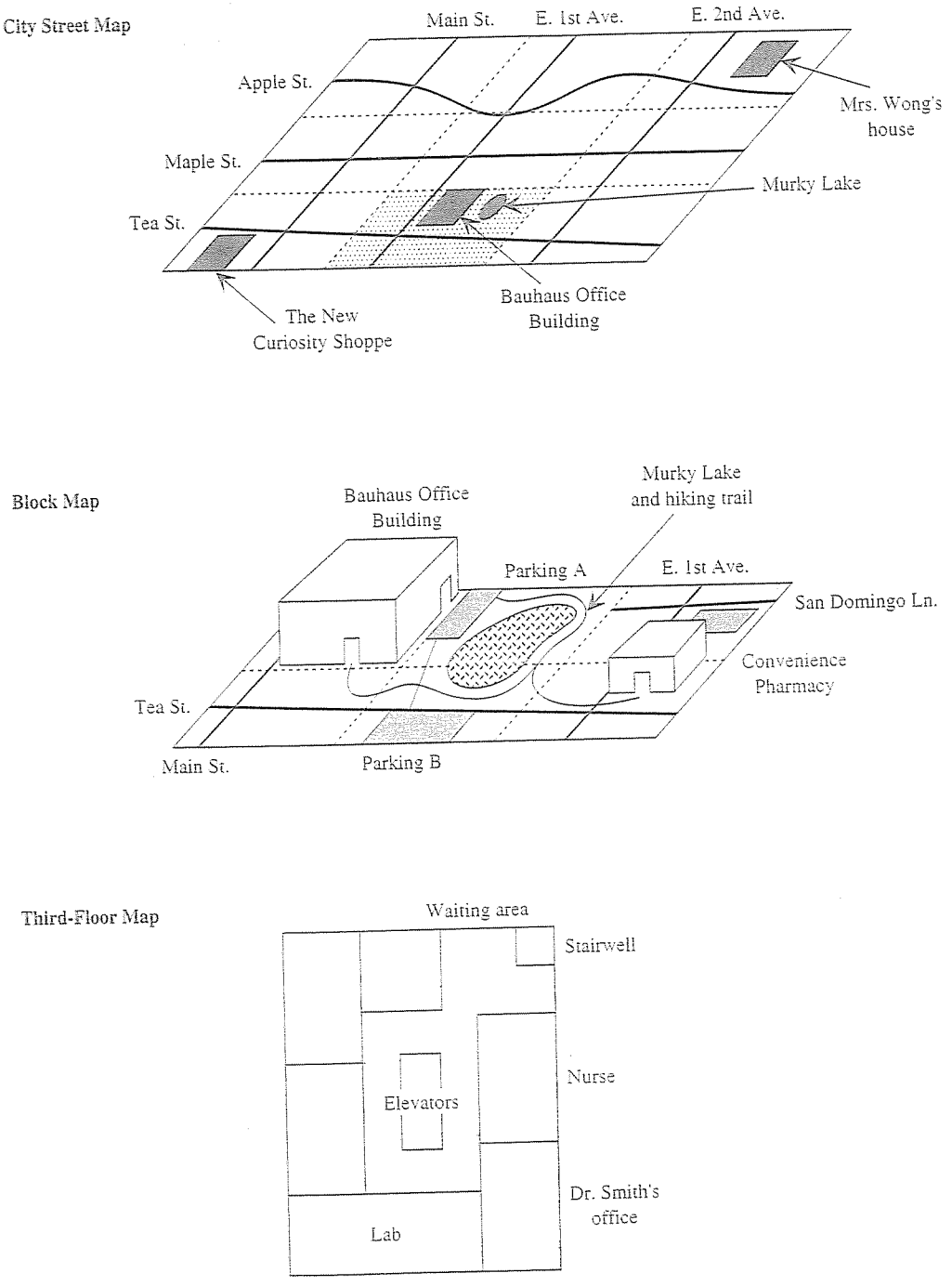

FIGURE 5.4. A multi-level spatial representation.

The most detailed level in our example database is the floor-plan level. Figure 5.4 shows the map of the third floor of the Bauhaus Office Building. This map shows the main places and routes that a visitor walking into the building would want to know about, such as the location of offices, stairways, and waiting rooms.

Consider a route-planning problem where Mrs. Wong is going to Dr. Smith's office for her first appointment. For all practical purposes, the city street map is enough to indicate that the Bauhaus Office Building is too far away for her to walk to it. It is also enough to plan a drive along East 2nd Street and then Tea Street. As Mrs. Wong approaches her destination, it becomes interesting to know about parking data on the intermediate level. She can select one of the two parking lots near the lake. Once she is inside the building, information from the floor-plan level becomes interesting, such as the location of Dr. Smith's office, the elevators, and the stairwell.

The use of multiple levels partitions the concerns of interest in spatial planning. Economy arises from being able to answer common questions using simple methods and "good enough" approximations, descending to greater levels of detail only as needed. In such an arrangement, the amount of detail facing a problem solver at different stages of the process remains roughly constant. When the problem solver is viewing the big map, many details are omitted so the number of objects in the view stays low. At each stage when the problem solver switches to a more detailed map, the region of interest is itself smaller so the total complexity stays low even though more details are shown in the smaller region. By reducing the area while increasing the detail, the problem solver manages the complexity of the view.

Multi-level representations require provisions for annotating the identity of objects across multiple levels. We need to know which symbols at different levels refer to the same physical object. It is sometimes convenient to omit information at a detailed level that has already been recorded at an abstract level.

Using a recursive region representation like a quadtree is not the same thing as having a hierarchical, multiple-level representation. The defining characteristic of a multi-level representation is that it represents different types of objects at each level. Typically, the highest levels in a multiple-level representation are the most abstract and the lowest are the most detailed. A quadtree representation provides a balanced way of allocating storage when objects are distributed unevenly. Each level in Figure 5.4 could be represented using a quadtree database.

5.3 *Reasoning about Shape*

So far our examples of spatial reasoning have emphasized maps and reasoning about routes. Within the context of maps, objects have been represented simply as being at located at points or as being contained within bounding boxes. We have not considered how object shapes could be described.

There is a vast literature about the representation of shape. We can give only a sketch of the ideas behind the main approaches. There are many possible representations of shape and the choice among representations depends, as usual, on the nature of the reasoning task. Figure 5.5 gives examples of four distinct approaches to the representation of shape:

- *Volume sweeping*. In the first example, a three-dimensional shape is described as the volume swept out by moving a planar figure along a curve. The curves may be described in any number of ways, such as piecewise linear or polynomial expressions. This approach is