May 1987

A Computational Treatment of Locative Relations in Natural Language

Ellen M. Hays
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-87-31.

A Computational Treatment of Locative Relations in Natural Language

Abstract

This paper discusses a system for talking about objects and spatial relations. The work was done in the context of a project called Landscan, for Language-Driven Scene Analyser. The system takes questions in natural language about a partially analysed image of a scene, extends the analysis of the scene as necessary, and responds with information about the objects it contains. Image processing and reasoning about the scene are guided by the input query. Landscan comprises (1) a vision system, which is responsible for image processing and object recognition, (2) a language processor, responsible for understanding the input queries, and (3) a reasoning agent, to determine what is already known or knowable about the subject of the query, to formulate requests for data to the vision system as necessary, and to compile those data into meaningful answers.

This report is concerned with the last two. Since most queries in this context concern objects and their spatial relations, it describes a computational treatment of Herskovits' work on locative expressions, and evaluates the usefulness of Herskovits' approach for this system. It also proposes a general design for the reasoner/interface, outlines the protocols required for the language and vision systems to interact with it, and points out aspects of the project needing particular attention. The very ambitious scope of the Landscan project has naturally made it difficult to do more than point the way to further exploration of many issues.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-87-31.

A COMPUTATIONAL TREATMENT OF LOCATIVE RELATIONS IN NATURAL LANGUAGE

Ellen M. Hays

MS-CIS-87-31
LINC LAB 58

Department Of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

May 1987

Acknowledgements: This research was supported in part by Air Force grant AFOSR F49620-85-0018, DARPA grants NOOO14-85-K-0018, NSF-CER grant MCS-8219196 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

A Computational Treatment of Locative Relations in Natural Language

Ellen M. Hays
Department of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia PA 19104-6389

May 1987

Abstract

This paper discusses a system for talking about objects and spatial relations. The work was done in the context of a project called Landscan, for Language-Driven Scene Analyser. The system takes questions in natural language about a partially analysed image of a scene, extends the analysis of the scene as necessary, and responds with information about the objects it contains. Image processing and reasoning about the scene are guided by the input query. Landscan comprises (1) a vision system, which is responsible for image processing and object recognition, (2) a language processor, responsible for understanding the input queries, and (3) a reasoning agent, to determine what is already known or knowable about the subject of the query, to formulate requests for data to the vision system as necessary, and to compile those data into meaningful answers.

This report is concerned with the last two. Since most queries in this context concern objects and their spatial relations, it describes a computational treatment of Herskovits' work on locative expressions, and evaluates the usefulness of Herskovits' approach for this system. It also proposes a general design for the reasoner/interface, outlines the protocols required for the language and vision systems to interact with it, and points out aspects of the project needing particular attention. The very ambitious scope of the Landscan project has naturally made it difficult to do more than point the way to further exploration of many issues.

Chapter 1

Introducing Landscan, and talking about place

1.1 Introducing Landscan

The study of natural language processing is motivated by a number of diverse concerns, prominent among them a desire for access by "naive" (untrained) users to various kinds of computationally derived information, whether stored in a database, static or dynamic, or available from sensors monitoring a physical process of some kind. Landscan (Language-Driven Scene Analyser) was conceived as a framework for studying a special case of the last of these: from data gathered using stereo-mounted cameras trained on a model of a city block, it was to reason and answer questions about the objects identified in the scene and the spatial relationships holding between them.[1] Further, it was to be query driven. That is, it would gather and process these data only in response to a user's questions, and, most important, information in the questions themselves would guide both reasoning and visual processing.

1.1.1 Theoretical motivations

A number of research aims are embodied in the project, in machine vision, in language processing, and in the design of a control structure for an interactive system integrating linguistic and visual components. Among these aims are:

- to explore the difference between what can be learned simply from looking at a scene and what must be derived (inferred) from a combination of incomplete visual information and previously-compiled knowledge about objects and spatial relations;
- to examine the question of how and to what extent an input query can be used to guide image processing; and
- to look at how the system's reasoning capacity can constrain the amount of image processing needed; if information requested by a user can be inferred from data already available to the reasoner, or with a modicum of additional data, requests to the vision component can be minimized.

[1] Foundations for the project, especially with respect to object recognition and aspects of the image processing, were laid by Bajcsy, Joshi, Krotkov, and Zwarico [4]. Additional low-level image-processing work was done by Liebman. A technical report by Zwarico [27] describes an interesting approach to recognizing objects and representing the scene.

Certain strategies suggest themselves in light of these interests: for one, the reasoner might create and maintain its own "mental map" of the scene from data returned by the vision system. Thus data previously gathered, whether directly needed for answering a question or not, would be on hand and available to be consulted and reasoned about in the course of subsequent exchanges.

The disparity between what is abstractly known about objects and what can be seen requires the ability to reason from indirect evidence. For instance, the information as to how many stories a building has is not directly ascertainable: the floors are not visible from the outside, and thus their number must be approximated either from the overall height of the building or from something like the number of rows of windows visible. Both of these methods are of course inexact, but they do offer reasonable approximations to an answer, which will often be all that is expected. The point is that a reasoning agent needs to be aware that a piece of information bearing on one of those things could be used in answering such a question.

Another strategy involves asking the user for clarification or guidance when either no description of a queried object or relation is contained in the system's world knowledge base or more details are needed to determine exactly what information is being asked for. For example, meanings for some input expressions might have to be defined by the user before processing of the request could begin. If one were to ask, "Is the statue beside an old building?" and the system didn't have a definition for *old*, it would perhaps want to ask the user to characterize *old* with respect to buildings, that is, to indicate what visual characteristics the system should look for (steeply pitched roof, chimneys) to make such a determination.

1.1.2 The structure of Landscan

Landscan has three major components: a natural language processor, to accept and process questions in English about the scene; a vision system, which works from low-level image processing "up" as far as recognition of objects and calculation of simple geometric information such as size and distance; and a reasoning module, to act as an interpreter between those two, "translating" a representation of the input sentence into a call to the vision component to return specific data regarding the contents of the scene, and then composing those data as needed into the more abstract conceptual units in which the user's request is framed.

Landscan as a comprehensive interactive question-answering system is obviously rather ambitious in its scope; the project touches on literally every thorny question in artificial intelligence research. The efforts of project participants have therefore been directed at dealing with some of those questions before work on the system as a whole continues.

1.1.3 Problems

Knowledge representation

As will be seen at a number of points in the present paper, the rock on which this work, like so many endeavors in the field of artificial intelligence, threatens to founder is the matter of representing knowledge adequately. In fact, it is quite difficult to say either (1) what we know, that is, the extent and depth of our knowledge, or (2) what aspects of it are brought into play in which situations (and would thus need to be represented in a system simulating a particular kind of human reasoning), or in particular (3) how to effectively encode and make use of what we do succeed in identifying as necessary to whatever kind of reasoning we want to do.

Take as an example knowledge about salience. Understanding and answering questions about objects and their spatial relations seems to require knowledge people clearly have about which parts or features of an object are salient in different situations [12, ch.6], but what is salient varies a good

deal with the context. For example, the top of a table is salient (and thus critical to whatever computation we are performing) when we are considering what might be resting on it, while one particular edge is salient when we are asking how near it is to the wall. So what is salient about an object with respect to one spatial relation will oftentimes not be with respect to another, or it may depend on what other objects are in the vicinity (or in the linguistic context, in the case of a conversation about objects and spatial relations). Compounding the confusion is the fact that the meaning of salience itself is somewhat variable; it may be related to function, or to physical (that is, visual) prominence, or to some notion of importance unrelated to either of these.

If it is the case that we have at our command, and need, the kind of knowledge about salience implied by the foregoing with respect to all the objects and relations we encounter in our everyday dealings with the world, clearly the quantity of such information required overall for reasoning about even the most mundane locative descriptions is staggering, not to mention difficult to encode. And this is in turn only a small fragment of the knowledge needed just to think and talk about a handful of objects in a picture.

Zwarico predicted this problem, and hinted at its severity, in her earlier work on Landscan [27]. No one, she noted, "has yet proposed a means of encoding the linguistic data which must be known about the objects in order to use them correctly in natural language utterances" (p.42). Approaches have been made, of course, including work by Bajcsy and Joshi [3], Waltz [23,24,25], and Talmy [20,21,22], among others, suggesting ways of addressing the question. A recent book by Mellish [17] takes an interesting approach to evaluation of referring expressions in verbal descriptions of mechanical objects that may be relevant here. Work has also been done in the vision field on representing 3-D information recovered from images, cf. Herman and Kanade [11], for example, though that work is concerned with representing not linguistic but spatial information.

Machine vision

Work on the vision component of Landscan has had some success; an interesting start has been made on designing an intelligent algorithm for isolating and identifying objects in this domain. But since some of the low-level processing being used was felt to be inadequate to the demands imposed by the larger project, Helen Anderson, who is responsible for that part of the system, has recently concentrated primarily on refining the strategies used for edge detection and segmentation. A recent technical report [1] contains an explanation and summary of the edge detection work, as well as outlining her strategy for object identification. The segmentation work is the subject of a forthcoming report [2].

Understanding locative expressions

One problem that surfaced early in my own work on the language processor for Landscan was the lack of an effective method for handling locative expressions, which occur often in conversation about a scene and the objects in it. Just as Anderson deemed it necessary to go back and deal with some of the inadequacies in the processes underlying object recognition, I have ended by giving most of my time and attention in the development of the language processor to understanding locative expressions, in addition to considering the question of how work done by the language processor could contribute to the reasoning ability of the system as a whole. The idea was that if the program could not "understand" the expressions used by a person to describe the locations of objects in the scene, it could not hope to accurately pick out the objects referred to, or comprehend what relations between them were being talked about, or answer correctly. If it could, on the other hand, it was hoped that that would both facilitate dealing with the user's requests for information

and, perhaps, elucidate some of the conceptual mechanisms at work in our own reasoning about objects and space.

The basis for this part of my work is the thorough examination of locative expressions recently completed by Annette Herskovits [12,13]. Although her analysis of the subject is far more comprehensive than is either necessary or appropriate for use in Landscan, it was clearly an excellent model with which to approach the question. In the end, it has proved useful in a more general way: in addition to guidance for understanding locative expressions, her work provides a conceptual framework for thinking about objects in space that I believe will be valuable to the Landscan system as a whole.

1.2 Talking about place

In many ways, spatial expressions are an ideal domain for thinking about the connections between language and the world; we have before us, so to speak, on the one hand the physical (or at least the visually perceptible) world, and on the other our linguistic mechanisms for referring to that world, objects in it, relations between them. Theories of how such referring works can be tested in a straightforward manner: does the expression being used adequately convey the situation? Is it ambiguous? If it fails, can we see how and why it does so?

One important point brought out by even a cursory glance at this subject is that we think and talk about our own rather idiosyncratic perceptions of it more than about the world itself; our conceptions of objects and spatial relations, as reflected in language, are highly idealized and of course much simplified. In fact, what they seem generally to reflect is a sort of "naive physics" [10] perception of the physical world. As Herskovits says, "The common-sense view of the physical world underlies our perception of it and every linguistic description of its physical aspect" [12, p. 27].[2]

For instance, we have no embarrassment about referring to a star as being located "to the left of the pine tree", in spite of the obvious facts that the star is millions of miles away and a pine tree has no left side, in the normal sense. We assign sides to the space surrounding inanimate objects (through a process Herskovits describes very clearly), and we designate, by implication only, a plane on which the star and the tree are seen side by side, to facilitate speaking (and, perhaps, reasoning) about these objects and the spatial relations in which we find them.

Herskovits, in her book *Language and Spatial Cognition* [12], investigated in depth the use of locative prepositions in English; the conclusions she reached, which I will represent as accurately as I can in the limited space of this paper, seem to me intuitively sound, though she makes no claims of formal provability for them. She considers appropriateness conditions for the use of locative prepositions from a number of different perspectives, before exemplifying her constructs in a close look at what she calls the "basic topological prepositions": *in*, *on*, and *at*; and the "projective" prepositions, such as *to the left of* and *in front of*, which may rely for their meaning on our knowing the observation point from which the area being referred to is perceived.

My intention in this work has been to try to implement in the context of a particular language processing program some of Herskovits' ideas, in spite of her own insistence that the "algorithm" she proposes is not intended or expected to be computationally complete, nor, perhaps, even realizable. She says in her preface that she herself deliberately avoided implementing her theory in a program, that "writing a program would have required simplifying to such an extent that it would have

[2] On the other hand, she also points out that this naive view of the physical world may not always be reflected directly in language: "There are abundant examples of divergence between this fundamental description of the world and the conceptualizations revealed by language"[12, p. 28].

contributed nothing to a demonstration of the validity of a delicate semantic analysis" (p. ix).[3]

I therefore propose not to demonstrate the validity of her analysis, though I believe it to be fundamentally valid, but rather simply to explore some of the problems encountered in an implementation attempt and speculate on what they might mean. Practically speaking, an antecedent intention, of course, was to look at the question of whether understanding locative expressions in the way proposed by Herskovits could facilitate the dialogue between user and system envisaged for Landscan. I believe that it will be seen from Chapters 3 and 4 of the present paper that it can. And although her work, as I have indicated, goes far deeper into the question of a cognitive model of locatives and how they are used than is necessary to my program, it was a blessing to have at my disposal, so to speak, a far more complete analysis than I really needed, and so to be able to choose those aspects of her account that were most useful to me. The question of whether this selective approach to her paradigm is in fact valid will be considered later. For the moment let me continue by describing her approach.

[3]Since the book *Language and Spatial Cognition* is the basis of most of the work discussed herein, I will henceforth identify quotes and references to specific parts of it by page or chapter number only.

Chapter 2

Theory and algorithm

Despite her repeated reminders that the "algorithm" does not and emphatically is not meant to constitute a plan for a real computer implementation, Herskovits has supplied, in broad outline and in some detail (though there are critical bits of detail missing, as will be noted), a very thorough description of the elements needed to interpret or, in her terms, to "decode" locative expressions. They consist of a general procedure to be followed, a predicate representation for the fully disambiguated expression, and a reasonably complete set of guidelines for building that representation, or rather for constraining the many possible ways of building it.

However, she warns, "these pages will offer only a first characterization of the underlying representations needed, of the results to be obtained, of which knowledge is needed for which task, and of the fundamental abilities the whole process presupposes. . . . In discussing what must be computed, I will occasionally assume some operations performed in some sequence, where it seems helpful to introduce some order in an otherwise unwieldy complexity, but this is not to be taken as an actual plan for computation" (p. 97). I will have more to say in Chapter 3 about the sequence of operations.

2.1 Theory

What follows is a brief explanation of the elements in Herskovits' account, with an attempt to show where each comes in to the understanding process, and a description of the "algorithm" itself, as outlined in the chapter of her book just cited on decoding and encoding locative expressions.

2.1.1 Figure and ground

Central to this approach to locative expressions is an understanding of what, in fact, is being represented. Herskovits follows Talmy [20] and others in seeing the objects in a locative prepositional phrase as being in a figure/ground relationship, that is, one object (the reference object) provides the ground against which the other (the located object) stands out as a figure. For example, in "the house near the stream", the location of the stream is assumed to be known or knowable to the hearer, and thus the location of the house is derivable from a simple two-place relation `near(house,stream)`. Note that in this example, as in most uses of locative prepositions, the precise location of the house is not specified linguistically; we know only that it is somewhere in the area (possibly very large) that can be said to be "near the stream".

In some locative expressions, three objects are involved. Typical of these expressions are words like *between* ("the house between the woods and the stream") and many projective prepositions, in which the point of view of the speaker is a necessary parameter in calculating the location being

sought (though it is rarely mentioned explicitly): "the cat is in front of the tree" normally demands that we know from which vantage point the tree is being considered. Herskovits refers to the observer here as an indexical variable that is free in the normal situation type (described in the next section) and will be bound by the particular context (p. 25).

An additional complication with regard to these projective expressions is the fact that there are two ways of referring to the sides (including front and back) of objects; "the tree to the left of the house" can refer either to the area at the left side of the speaker's field of vision or to a left side imputed to the house itself, figuring from the side designated as the front (ch.10).

2.1.2 Normal situation types

Herskovits sees each locative expression as having a **normal situation type**, that is, as being defined by "a set of characteristic constraints . . . that must hold for the expression to be used truly and appropriately under normal conditions" (p. 20). An example of the sort of constraint she has in mind here is the fact that normally the use of the preposition *near* requires that the reference object be at least as immobile as (and preferably much more fixed than) the located object—we would say "the bicycle is near the house", but not normally "the house is near the bicycle".[1] Another such constraint has to do with conventional orientation of a reference object; a striking example she gives of this is the inappropriateness of the use of *in* to refer to an object inside a bowl that is upside down. Most speakers will use *under* in that situation, precisely to draw attention to the unusual circumstance of the container's not being in the position in which it can normally contain things (p. 23).

Unfortunately, there does not appear to be any obvious way to formally encode this kind of information; Herskovits herself does not propose one, and my own approach has been merely to designate a set of tests pertinent to each individual preposition to be executed whenever that preposition is encountered. It seems clear, in any case, that these constraints are more relevant to choosing the appropriate preposition to use to convey some relationship between objects than to interpreting an expression used by someone else, where by default one assumes the speaker is adhering to Joshi's revised maxim of quality [14] (an amendment to Grice's cooperative principle [9]) by avoiding a misleading implicature about the situation.

2.1.3 Purpose

The prototypical purpose for the use of a locative expression, according to Herskovits, is to locate an object, that is, to answer the question "Where is X?", by providing enough of a constraint on the location of X that it can be easily found by the hearer (but not so much as to be unnecessarily specific, again in line with Grice's cooperative principle.)

Of the non-prototypical purposes, the most common is certainly that of identification; in a sentence such as "The tree in the corner of the yard is a linden", the locative prepositional phrase functions as a specifier on the noun *tree*.

My own work with locatives has been concerned with these two purposes only. All the others she mentions (e.g., describing a located object, where the location is irrelevant, as in "the man in the mac", which seems to me hardly a locative expression at all) have meaning only in non-geometric domains well outside the scope of Landscan. I have also followed Herskovits in dealing only with static prepositions, avoiding entirely the dynamic uses, for the simple reason that the domain contains no notion of change over time. Thus, my program handles the preposition *across* as it is used in "the house across the street", but not as in "the dog trotted across the street", that

[1] This example is due to Talmy [20].

is, it deals only with the case where a phrase like "across the street" further specifies an object, rather than an action.

2.1.4 Ideal meanings

Ideal meanings are the closest Herskovits comes to abstract definitions for the locative prepositions:

> The ideal meaning of a preposition is a geometrical idea, from which all uses of that preposition derive by means of various adaptations and shifts. An ideal meaning is generally a relation between two or three ideal geometric objects (e.g., points, lines, surfaces, volumes, vectors)—in fact, ideal meanings are usually those simple relations that most linguists and workers in artificial intelligence have proposed as meanings of the prepositions. These relations play indeed an important role, but as something akin to prototypes, not as truth-conditional meanings. (p. 39)

An example of an ideal meaning is the one proposed for *in*, *viz.*:

> inclusion of a geometric construct in a one-, two-, or three-dimensional geometric construct

Since this "definition" is intended to be as general as possible, that is, to comprehend uses of *in* as diverse as "the cat in the tree" and "a crack in the ceiling", it must of course be considerably constrained and refined to produce a coherent interpretation for those two examples, or in fact for any other locative use of *in*.[2] The necessary refinements will come from geometric description functions, from certain pragmatic generalizations, and from descriptions of the diverse senses of prepositions Herskovits calls "use types". Before elaborating each of these, I will introduce a quasi-formal notation (taken from Herskovits) for locative expressions.

2.1.5 Predicate representation

Herskovits' proposal for representing these relations begins with a simple predication of the sort *prep(X,Y)* or *prep(X,Y,Z)*, where X is the located object and Y and Z the reference objects. The locations of X, Y, and Z are then subject to further specification by means of geometric functions that can be applied to them to produce the more precise characterizations needed to determine their place in the construct (and, additionally, to bring them into line with the argument types required by the ideal meaning of the preposition itself—more on that later). As an example, the representation for "the bird in the tree" in the proposed notation would be developed from `in(bird,tree)` to become:

```
included(place(bird),interior(outline(visiblepart(place(tree)))))
```

where the geometric description applicable to the tree is the result of nested applications of the "elementary geometric description functions" (next section) `interior()`, `outline()`, `visiblepart()`, and `place()`, all of which must be defined in some axiomatic or algorithmic way.

The formal schemata for these are:

$$\mathrm{A}(S_I)(G_1(O_1),G_2(O_2))$$

[2] It is not intended to comprehend uses such as that in "there's been a change in plans", nor, except by analogy with physical containment, the uses that occur in the present sentence.

and

$$\mathrm{A}(S_I)(G_1(O_1),G_2(O_2),G_3(O_3))$$

where S_I is a predicate derived from the ideal meaning of the preposition, A the "tolerance shift", a relaxing of the ideal meaning to handle the not-quite-normal uses or extensions of accepted meanings we allow for prepositions, and $G_i(O_i)$ the geometric functions applied to the various objects in the expression.

2.1.6 Geometric description functions

> To understand the use of locative prepositions, one must assume that geometric descriptions are mapped onto the objects by a process of geometric imagination, a mapping accomplished by **geometric description functions**. (p. 57)

The basic geometric description function for any object X in space is `place(X)`; this describes the entire location of the object. But since we often mean some part or idealization of an object when we use it in a locative expression, we are, in her terms, applying further geometric description functions to it. For instance, if an object is on a table, it is in fact on the top of the table, and thus its location is actually `overside(place(table))`, to use her name for the function that returns the top surface of an object. Clearly, functions can be nested to any depth, though in general one or at most two removes from `place()` of the object is enough to refine the meaning of its location enough for its contribution to the locative expression to be clear.

Herskovits classifies the geometric description functions according to how they modify the meaning of `place()`; among the classes she designates are:

- parts, such as `surface()`, `overside()`, or `edge()`;
- idealizations, such as approximations to different geometric objects (for example, `ptapprox()`, approximation to a point, which occurs frequently with respect to certain prepositions);
- projections, on the ground, or on some other plane at infinity; this is what is involved in the "star to the left of the tree" example in Chapter 1.

A fully instantiated representation of a locative expression, then, would be something like this for "the car is in the corner of the parking lot":

```
included(base(place(car)),area_from_vertex(corner(place(parking_lot))))
```

Here the geometric functions applied reflect certain "facts" about this preposition: that when we refer to a three-dimensional object as being *in* an area, it is the base of the object that is salient, and that a corner, which is viewed as a point, must be conceptualized as an area to be seen as containing any object of greater dimensionality than a point.

2.1.7 Use types

Each preposition has a catalogue of **use types** attached to it; this is a set of pattern-interpretation pairs, such as:

> N(spatial object) in N(container): *spatial object in container*

(where N(x) is a noun phrase of category x), which designates one of the conventional "senses" of *in*, namely, that of (partial or total) containment of one object by another.

> . . . all such conventional facts of use—facts that are neither determined by the ideal meaning of the preposition and the meanings of the subject and object of the expression, nor pragmatically inferable—will have to be somehow specified in the lexicon, as characteristics of additional senses of the preposition or of idiomatic forms. . . A collection of use types will be attached to each ideal meaning of each preposition. (pp. 86-87)

Herskovits enumerates eleven use types for *in*, of which I have been able to find only three that seem applicable to the Landscan domain, *viz.*:

- physical object in outline of another, or of a group of objects ("a cat in a tree")
- accident/object part of physical or geometric object ("a curve in the road")
- spatial entity in area ("the trees in the park")

As she notes, use types "do not fully specify tolerance and geometric descriptions", that is, a number of additional constraints must be applied before an interpretation based on one of the available use types can be determined.

2.1.8 Pragmatics

In Herskovits' view, four main properties of objects and their spatial relations need to be considered with respect to pragmatic constraints on the use of locatives: they are salience, relevance, typicality, and tolerance. She has formulated several "near principles" (her term) regarding the way these properties influence meaning, but says of the principles, "These are not predictive; they embody necessary but not sufficient conditions for the appropriateness of a certain use, and are formulated in terms of factors for which we lack a formal account" (p. 73).

She does propose a general explanation, based on the notion of salience, for how synecdoche works, and also makes the use of idealizations plausible, but her point that there is no formal account of these things is well taken. It is clear that they are entirely pertinent to our endeavor; the problem is how to formalize and use them.

2.2 Algorithm

In Chapter 8 of her book, Herskovits outlines a procedure that looks very much like an algorithm, with many disclaimers about its completeness. I have reproduced it here in a form that makes it look a good deal more like one.

Overall, the decoding of locative expressions under normal conditions seems to be composed of these two steps (p. 105):

1. "construct the normal situation type(s) associated with the spatial expression," and
2. "given such situation type(s), exploit the particular context to further specify the normal interpretation."

The interpretation, in turn, "consists of a set of constraints, which can be subcategorized into those constraints that apply directly to the scene and those that apply to the context" (p. 107). Of these, the constraints on the scene ("selection restrictions, allowed spatial relations between the objects, constraints on indexicals, and constraints on geometric description functions") are considerably easier to codify and test for than constraints on the context ("constraint on purpose, highlighting of a background element, particular conditions on context") (p. 91).

This is what I understand her general "algorithm" for step 1 of the process outlined above to be, a bit simplified, and rephrased somewhat to imitate computational usage:

```
WHILE there are untried use types of the preposition under
consideration:
    1. "pick a use type"
    2. "hypothesize plausible geometric descriptions and tolerance"
           a. apply pattern and selection restrictions on noun
              phrases in the expression
           b. apply constraints on geometric description functions,
              which come from:
                    i.   use types
                    ii.  pragmatic principles
                    iii. object knowledge
    3. IF "the resulting geometric meaning and other instantiated
       use type constraints may be true, given our knowledge of
       objects"
       THEN "a normal interpretation can be constructed"
       ELSE "the use type is rejected"
ENDWHILE
```

In elaborating the second step of the general approach, Herskovits speaks only of obtaining "an interpretation more specific than the normal situation type." She suggests that "one will usually (a) identify the referents of the subject and object, (b) choose between possible geometric descriptions, (c) specify the tolerance, (d) assign a value to any indexical, and (e) draw additional inferences" (p. 112).

Parts of this second step, in particular (a), (c), and (d), are clear enough, but they will be needed elsewhere in my system, in particular by the reasoner, in collaboration with the vision system (this will be discussed in Chapter 4). The choice between geometric descriptions (b) is being done within the cycle described as step 1, and the last, (e), seems to me too vague to implement in an application as specific as mine. Accordingly, I have not attempted even to specify any but (b) in terms of computational tasks in this program.

In general, my approach to interpreting locative expressions, which will be elaborated in the next chapter, has been to try to incorporate as much of the required constraint testing and reasoning as possible in simple procedures that rely on relatively *ad hoc* choices and very informal heuristics.

Chapter 3

Putting the theory to work: implementation

The challenge in implementing a design like this one is to capture at once the general intention or spirit of the theory and as many as possible of the specifics proposed therein, and of course to produce the desired result, in some verifiable form. Difficulties arise on the following issues:

- Representing knowledge: Herskovits' plan calls for testing for such relative intangibles as salience of parts of objects, relevance of different kinds of conceptualization to a given situation, relative typicality of different shapes, functions of objects, ways of idealizing geometric entities, and so on. I believe that representing these and other such things is both critical to an implementation of her ideas and exceedingly difficult, perhaps impossible to do adequately.
- Verification: the basis for judgments concerning the correctness or appropriateness of use of locative expressions is extremely unclear. It is hard to see how such judgments can be made in a way that is not largely arbitrary.
- Sequence: the order in which the various tests are to be performed is not completely specified, nor is it obvious that it matters much in many cases. Indeed, I am inclined to argue, after careful examination of the paradigm, that the description of constraints to be applied in interpreting locative expressions is essentially declarative, rather than procedural, and thus the testing process can be sensibly ordered according to considerations of computational efficiency and effectiveness. If that is true, it would also be interesting to examine the question of how much of this testing might be performed in parallel. I will have more to say on this point in the last chapter.

Most of the inadequacies in the program to be described stem from one or more of those three problems. A few others are the result of trying to implement the theory within the Landscan domain. Still others will undoubtedly be the result of my own uncertainty about how to make the best use of the theory in a computational context. A more comprehensive analysis of the implementation is undertaken at the end of this chapter.

3.1 The program

What follows is a description of my program, which takes as input a sentence containing a locative expression (either a noun phrase consisting of a noun and a locative prepositional phrase, or a noun phrase followed by a verb of being and a locative adjectival complement), extracts the preposition

and the nouns that are its arguments, and examines that locative relation to determine its probable interpretation. It returns a completed predicate representation (section 2.1.5) of the interpreted expression and indicates which use type was invoked in forming it.

An outline of the process followed in the interpretation of a locative expression appears at the end of this chapter, along with a few examples of the program's output.

3.1.1 Parser

Parsing of the input sentence is accomplished by means of a simple definite clause grammar (DCG), written in Prolog [19]. For the moment, the DCG returns a term in which the preposition is the functor and the nouns to which it applies are the arguments. For the input sentence "Is there a tree in the yard?" the parser would return `in(tree,yard)`. Parts of objects are combined with the objects they modify in a similar predicate relation which is then nested in the larger structure: for "a house at the edge of the park" the parser returns `at(house,edge(park))`.

The motive for using a DCG is that it provides a particularly straightforward way of extracting the preposition and its arguments from the input. Since only the interrogative forms of the uses of locative expressions discussed in section 2.1.3 are expected as inputs to this program, it is safe to take a noun phrase preceding the preposition to be the located object (whether or not there is an intervening copula),[1] and the one following it to be the reference object.

Clearly, a number of important linguistic matters are being dodged here: the parser returns no information about quantification, or even about definiteness of reference; the default assumption is that the objects referred to exist, or, in the case of the located object, its existence may be what is being queried with the locative.

Other information in the sentence will of course have to be retained for examination when the system is fully operative (see section 4.2.4). Questions like "How many trees are next to the driveway?" should not be difficult for the parser to handle; the representation would in fact need to be only marginally more complex than the one outlined by Pereira and Warren in 1980 [19, pp.252-53], though later developments in logic grammars [7,15,16] suggest ways of making the representation more semantically sophisticated and thus probably more useful for the integrated system.

3.1.2 Object checking

On receipt of the representation of the input, the control routine pulls out for examination all of the use types on file for that preposition. For example, each use type currently available for *in*, as outlined in section 2.1.7, is stored as a predicate with the preposition, an index, and a verbal description of the use type, corresponding to the interpretation in the pattern-interpretation pair described in the same section, e.g.:

```
use_type(in,1,'physical object in outline of another, or of a group of objects').
use_type(in,2,'accident/object part of physical or geometric object').
use_type(in,3,'spatial entity in area').
```

Attached to each use type is another data structure that corresponds roughly to the use type pattern, which contains very general selection restrictions for the objects that can appear in the X and Y (and Z, in the case of three-place prepositions) positions—in other words, that can occur

[1] That is, no distinction is made between, for example, "Is there a house on the corner?" and "Which house is on the corner?".

as the located and reference object(s), respectively, in that use of the preposition. The patterns corresponding to the use types just given for *in* are:[2]

```
pattern(in,1,physical_object,physical_object).
pattern(in,1,physical_object,group).
pattern(in,2,geometric_object,spatial_entity).
pattern(in,3,physical_object,two_d_object).
```

This is in accordance with Herskovits' remark that these restrictions constitute a first filter to be applied in determining which use types should be under consideration. The process does, in fact, eliminate all but one or two use types in most situations.

Another constraint is also being as it were passively tested at this point: the figure/ground relation "requires" (actually, strongly prefers) that the reference object argument to most prepositions be larger and/or more fixed than the located object. This preference is often uncovered by the object checking; for example, a number of the patterns call for geographic objects in the Y position (geographic objects are parts of the earth's surface—streets, sidewalks, parks, and so on), which are pretty much guaranteed to be as fixed as or more fixed than any located object. Where both objects are three-dimensional, of course, relative size or mobility has to be checked explicitly.

3.1.3 Filling in the templates

The next phase in the interpretation process consists of finding what I call a "template" for the preposition under consideration and filling it in with appropriate geometric functions. Templates encode a sort of ideal meaning for each use type, rather than for the preposition in general (the exception is the template for *at*, explained below). That is, the predicate is the one that figures in the ideal meaning proposed by Herskovits, but the geometric types of the arguments may vary with the different use types, reflecting additional constraints on the parts or idealizations of objects that must figure in the interpretation. Templates are stored with the preposition and use type index so that only templates relevant to the use types that have survived object checking are considered. The templates corresponding to the use types for *in* (shown above) are:[3,4]

```
template(in,1,included(area,area)).
template(in,1,included(volume,volume)).
template(in,2,included(geometric_object,geometric_object)).
template(in,3,included(surface,area)).
```

This filling-in process actually collapses several of the constraint tests mentioned in the algorithm: we are simultaneously getting the geometric types relevant to the use type, finding geometric functions that bring the innate geometric types of the objects into line with the types called for by the template, and instantiating the argument positions of the predicate that designates the ideal meaning with geometric functions that produce the desired type coercion. Let me explain this notion of "type coercion" before discussing constraints on the process.

[2]The two patterns for *in,1* reflect the disjunction in the interpretation (above) for this use type.

[3]The space in which something is included in use type *in,1* can be either two- or three-dimensional; the located object must be of the same dimensionality as the space.

[4]The type *geometric_object* is the subsumer of all specific geometric types, that is, *volume*, *area*, etc. In other words, this slot may be filled by an object of any dimensionality.

3.1.4 Type coercion

In several programming languages, it is possible to explicitly convert a variable of one data type into another type by a process called coercing or "casting". For instance, an integer variable whose value is 5 can be coerced to a real number variable whose value is 5.0. The process of converting the geometric type of an object (e.g., area, line, point) into another seems to me analogous to this, hence the use of that expression.

The ideal meanings of the prepositions, as noted in section 2.1.4, specify geometric objects that can figure as arguments to some predicate that "defines" the abstract or ideal meaning of the preposition. Thus, for example, the ideal meaning of *at* is "for a point to coincide with another". My template for all use types of *at* reflects that ideal: `coincide(point,point)`. In other cases, individual use types for a given preposition may call for different geometric objects in the various argument positions: for example, Herskovits makes the point that when we talk about a three-dimensional object in an area (e.g., "a car in the parking lot"), we are actually talking about the *base* of that object, a surface, not the whole three-dimensional object. (Indeed, it seems to be the case that in general the dimensionality of the reference object must be the same as or greater than that of the located object, which generalization is usually reflected in the templates.)

What this means for the application of geometric description functions to some object X, then, is that one or more functions must as it were "coerce" the innate type of X (i.e., the type returned by `place(X)`[5]) into the type called for by the ideal meaning. Specifically, in the case of *at*, the most frequently applied function is `ptapprox()`, which must be applied either directly to `place(X)` (assuming X is not already a point), or applied to some other function that has already been applied to `place(X)`.

Example: "the house at the corner" contains two objects whose innate geometric object types are *volume* and *point*, respectively. Thus for `coincide()` to apply to them, *volume* must be coerced to *point* in the case of the first object, but no coercion is necessary in the case of the second, since "corner" is a location, which is defined as returning a point. (A location is defined in terms of a single pair or triple of coordinates in two- or three-space; it therefore has no dimension.) That expression will normally be interpreted as:

```
coincide(ptapprox(place(house)),place(corner))
```

Note that if it is not possible to coerce the innate type directly into the one called for by the template (for example, there is no geometric function in this program that coerces *line* to *area*), it might be necessary to apply two functions to achieve the needed coercion. Herskovits gives a number of examples of this, but I was unable to think of any situation that could use it in my domain, so I have not implemented it. It should in theory be relatively easy to make the type-coercion process recursive in order to accomplish this.

3.1.5 Other constraints

Having found the template defining what I call the "top-level" predicate for a preposition, and having to hand the innate types of the objects under consideration (that is, the meaning of the "bottom-level" predicate, `place()`, with respect to each), we are ready to begin working from those two ends of the representation toward the middle, i.e., applying intervening functions in a way that is compatible with the use type and the objects in question. This is done by examining the geometric description functions that accomplish the desired coercion, of which there are usually

[5]The `place()` function was explained in section 2.1.6.

several, and determining which of them is appropriate in the current context by applying specific constraints.

Pragmatic factors

As noted in section 2.1.8, the pragmatic factors relevant to constraining the interpretation are salience, relevance, typicality, and tolerance. Of these I have tried to implement only a test for salience at this point, though the others might, in a very limited way, be amenable to some attempt at a treatment. Salience of parts of objects is looked at with respect to specific use types. Take for example the use of *on* in the case of an object resting on ("contiguous with and supported by") another; it is the top surface of the reference object and the base of the located object that are salient here.[6] We say, "The teapot is on the table", but the contiguity relation applies to the bottom of one and the top of the other, and that is what the geometric description functions mediating between the places of the two objects and the relation *contiguous* must reflect:

```
contiguous(base(place(teapot)),overside(place(table))) &
                supported_by(teapot,table)
```

Tests based on the remainder of the pragmatic factors seemed to me either dependent on knowledge that is difficult or impossible to capture or pertinent primarily to the encoding rather than the decoding of locatives; I will have further comments on these in section 3.1.7.

Object knowledge

A number of other facts that are known about objects are treated in the database, which is a KL-ONE-style [5] representation, with objects, geometric or physical, in a hierarchical structure, linked further by roles representing information about the functions that coerce one geometric type to another (and thus, by inheritance, which objects can figure in these operations).

In the absence of a representation for sets of objects, plural nouns are dealt with in line with my own observation that the meaning of the plural varies according to whether it is the X or Y argument that is a plural. If it is the located object (X), I assume that the user is asking a single question referring to the location of more than one object of the same kind (and thus tests applied in answering the question will consider each member of that group). If it is the reference object (Y), I assume that the objects named constitute a single unit, the object type of which is *group*, and that the location is to be found by examining the space defined by an outline of the group, that is, the equivalent of the convex hull.

For example, in a phrase like "the house in the trees" the meaning of *trees* is something like "the space delimited by an outline around all the trees", where in "the trees at the edge of the park" we can specify the place of the trees in general, but a response to a question about them will have to include testing for each that it is in fact at the edge of the park. These assumptions vary a bit with the preposition being tested; *between* (as in "a tree between the houses") is a special case in which it is necessary to ascertain whether some small number (probably two) of the reference objects were previously mentioned, in which case the plural can be interpreted as designating Y and Z arguments to the preposition ("a tree between house1 and house2").

Further knowledge about objects comes from the hierarchical structure of the knowledge base, that is, generalizations can be made at any appropriate level and the attributes inherited by subsumed objects. Objects are specified as to the geometric type they return (actually the type

[6] It might be as easily said that the base and top surfaces are *relevant* here, but in Herskovits' terms relevance "has to do with communicative goals, with what the speaker wishes to express or imply in the present context." (p. 76)

returned by `place(Object)`), and additional information is available about some of the particular features we use to reason about relative locations of things.

3.1.6 Results

As noted, the program returns a completed predicate for the locative expression, or more than one if it has been unable to fully determine which of two possible interpretations is correct, and an indication as to which of the use types for the preposition it used in building that interpretation. Judgments as to the appropriateness of the final results, and the choice between multiple interpretations, where they occur, have to be made by the user, at this point.

In this program I have used Herskovits' use types only, creating my own constraints, in defining the patterns and templates for each, on how an interpretation is built from them (following many of the ideas presented in her book). These more or less *ad hoc* constraint definitions have had the effect of making the relatively vague descriptions of the use types considerably more precise.

An outline of the program's behavior and several examples of its output are to be found in section 3.3.

3.1.7 Limitations of the program

One problem with implementing Herskovits' ideas in a program that does only *decoding* of locative expressions is that her account is an attempt to explain the various phenomena involved in the use of locatives, with little distinction between which of the constraints she discusses are relevant primarily to the decoding and which to the encoding process. Thus it was necessary, in addition to seeing how her constraints might be represented, to determine which of the many aspects of locative use she describes would actually be useful for decoding *per se*. It is not clear to me that I accomplished that, and I am thus not prepared to claim that my results are definitive on that count; in particular I wonder if a system designed to both decode and encode locative expressions might not make better and more integrated use of the many insights reflected in her book.

An example of this non-distinction I found particularly troubling concerns the pragmatic "near principles". Since for the most part the pragmatic principles exist to explain uses rather than to aid in interpreting them, I have not generally found it possible to implement them in this program. In the previous section I discussed a limited treatment of salience with respect to objects. The remaining factors covered in Herskovits' account, along with brief comments on implementing them, are these:

- Relevance, as noted earlier, deals with the communicative goals of the speaker and thus seems primarily useful to encoding locatives, that is, choosing which preposition to use in some situation. Of course the speaker's goals, if they can be identified, constitute a factor that could be quite useful in the interpreting process. I found relevance the least tangible of the notions dealt with, and the hardest to represent.

- Tolerance will, I think, properly be incorporated in the part of the system that tests the existence or truth value of the spatial relation described. If the appropriateness of the use of *near*, for example, is dependent on the size of the objects in the relation, then that constraint on a particular use of the word cannot be tested until the actual physical parameters of the objects and distance between them are being tested, in this case, by a call to the vision system for data to answer some question.[7]

[7]The need for this sort of reasoning motivates my proposal in Chapter 4 that the geometric functions used in the interpretation of a locative expression be part of the semantic representation passed to the reasoner, so that they

- Typicality seems to me a property best captured in a rule-based sort of representation of meanings of locatives that would also figure as part of the reasoner/interface in a system like Landscan. Such a representation would have to include certain facts of typical or default intentions (reference and located objects are assumed to be close together unless something is said to obviate that inference, for example) as well as facts about typical function, typical shape, and so on.

I have also neglected entirely a treatment of the notion Herskovits calls "tolerance shifts", denoted by the predicate A() (which means something like "approximately" or "almost") in the schemata in section 2.1.5, feeling that although the concept is an important part of Herskovits' account, it was simply too fuzzy to represent here. I believe that for the purposes of this program the effect of that neglect is not too serious; A() represents a degree of subtlety that is not necessary for understanding a locative expression, though it might need to be implemented elsewhere in Landscan, to figure, for example, in the definition used for testing whether a given relation holds with respect to particular objects.

Imposed by computing

In general, the limitations imposed by computing have to do with the difficulty of representing the subtle distinctions that are needed to determine whether an interpretation is appropriate, whether a given geometric description function should properly apply to an object, whether a sense shift (a shift "to another, conceptually close relation" (p. 40), exemplified by the use of *on* in "an apple on the branch", which differs from the normal contiguity and support meaning of *on* in that the apple is not above the branch, and the contiguity involves only one point rather than a surface) is plausible. All these matters are judgment calls, and while Herskovits has made fairly clear pronouncements, for the most part, on how *we* make them, it's not easy to see how the knowledge needed, be it facts or rules, is to be captured in a program.

In addition, of course, as was noted in section 1.1.3, the amount of detailed world knowledge required for this process is enormous, which inevitably puts a crippling load on computational resources with respect to both space and time.

Imposed by Landscan

Certain limitations on the implementation were imposed by the desire to stay generally within the Landscan domain. For one thing, the number of objects available to be talked about is very small, and distinctions between them of only the crudest sort. For that reason, I have allowed in my program's domain objects that seemed likely to be recognizable in the near future (*car*, *field*), as well as a few objects necessary to expand the range of spatial relations available for discussion, such as parts or adjuncts of buildings (*roof*, *chimney*).

In addition, since we have only one perspective on the scene (looking directly down at it), and since the system's ability to model height of objects is very limited at this time (height is being returned as a boolean value, that is, the top of an object is known to be either at or above ground level), reasoning about relative size and certain aspects of spatial location needed for many of the interactions we would like to model is either not possible or severely hampered. Again I have sometimes pretended that more information of that kind was available than is currently the case, for the sake of the conversation.

---

may be used in turn to guide the vision system in both object recognition and calculation of the parameters defining relations.

As an example of these constraints, the horizontal perspective needed for interpreting most projective prepositions is lacking; if we ask whether an object is *to the left of* another, we have the standard problem of whether that means to the speaker's left or to the object's left, assuming left and right sides can be imputed to the reference object, but in this situation the default meaning will have to be "to the viewer's left in the image plane", since the image plane imposes its own basic orientation on all the objects in it. To say that something is *in front of* another, though, can have only one meaning in this system, namely, to the side of the object previously explicitly designated as its front, since there is no space between the viewer/speaker and any object in the scene to which one could be referring.

Another more serious restriction imposed by the domain is that because conversations with Landscan are exclusively about geometric objects and relations, many of the use types formulated by Herskovits, and particularly many of the most interesting ones, either cannot occur at all or cannot be reasoned about because the information needed to test their validity is not available. Thus many of the distinctions her account is designed to handle between the various interpretations of a preposition are simply never needed here.

3.2 Analysis and conclusions

It should be noted that something like the implementation just described would be useful to any language-vision interface, not just to Landscan. The need for a fine-grained interpretation of expressions referring to locations of objects is not limited to a system like this, in which we intend to let the input inform the work of the system, though it may be particularly valuable here. In the next chapter I will indicate how the locative expression interpreter is to be integrated with the rest of the language processing; that, too, will be relevant to any natural language question-answering system for discussing objects and spatial relations.

One rather alarming question implied by the implementation described here is this: is the degree of subtlety and sophistication required by this program to handle locative expressions adequately also needed in respect of every other aspect of language? In other words, will we need systems this complex (or more so) only to deal with referring expressions, or with quantification, or tense, or what have you? If so, will the ideal of a language processing system that is in some sense "complete" have to be abandoned just for lack of space to store the needed information and time to process all of it?

In this case, of course, the full power of Herskovits' paradigm is unneeded; a cut-down version is quite adequate to cover the limited range of dialogue and topics that will be encountered here. But even that very restricted version is difficult to implement entirely. Herskovits' warnings about the complexity of the problem are not exaggerated. But the program does show up a number of interesting points, which I have classed as counts for and against trying to use a theory like hers to shape a natural language processing program.

3.2.1 Counts against

Herskovits' proposed "testing" of the interpretations of locative expressions is inherently a questionable concept; it is never explained how the judgment of appropriateness is finally arrived at (that is, after all the available constraints have been applied), except by recourse to subjective human opinion. Thus by definition a program that makes such "judgments" bases them on a series of *ad hoc* (though hopefully well motivated) pre-encoded decisions about how these things (prepositions, references to objects, spatial conceptualizations) are normally used. The pragmatic considerations are an example of all that is most uncapturable about the rules informing these judgments.

Herskovits herself says of this:

> This description of decoding calls upon some basic reasoning abilities which one must assume underlie language processing: checking consistency, selecting a best match, and drawing inferences. The modeling of these abilities corresponds to some fundamental research problems of artificial intelligence . . . (p. 111)

Such an attempt as this program is therefore inevitably an exercise in paring much that is important and meaningful from the theory, to bring the endeavor to a manageable size and shape. Whether too much has thereby been lost is moot; it may depend on how nearly we expect machine language processing to simulate human language processing, what degree of deviation we accept as unavoidable.

3.2.2 Counts for

The main argument in favor of using an approach like this must be that the account Herskovits has given of the use of locative expressions seems exceptionally sound and coherent, and although not intended for such a restricted application is sufficiently well conceived to behave reasonably well even under these very limiting conditions. Indeed, it is difficult to imagine an implementation of the entire theory, given the tremendous demands it makes on both cognitive and computational resources. So perhaps the most useful thing for a natural language processor to do is to use the parts that seem relevant, remaining as faithful as possible to the spirit and intentions of the theory, as I have tried to do.

Ultimately, my feeling about cannibalizing Herskovits' theory and applying it to a smaller and much less complex domain than it could in principle accommodate is that it is quite valid for this purpose, that is, as long as it is kept clearly in mind that the implementation does not represent the theory as a whole; it merely shows how far a selected subset of her rules and distinctions can still yield coherent and usable interpretations of the uses of locatives that do occur in this domain.

A more important question, from the point of view of my broader task, is how this approach fits in with the larger motivations arising from the needs of Landscan. My conclusion is that, suitably extended, it should work well, lending a degree of precision to our interpretation of the user's questions that will be enormously useful to the rest of the system. In addition, I believe that Herskovits' paradigm makes an important contribution to the problem of representing objects in space, quite apart from the linguistic considerations, one that has particularly interesting applications in the context of this project, or indeed for any language-vision system. I will have more to say on both of these points in the next chapter, on the future development of Landscan.

3.3 Outline and samples

3.3.1 Outline of the program

Here is a general outline of the program's behavior, which differs somewhat from the "algorithm" presented in Chapter 2, mostly with respect to how the various operations are organized. After reading the input sentence, the program:

- shows the preposition and its arguments, extracted from the input sentence;
- shows all the use types available for the preposition in question;

- looks at the pattern connected with each use type (in a few cases there are two acceptable patterns)—this is how the objects in the input are screened to see which if any of the use types they are compatible with;
- for each use type that has passed the object checking screen: looks at the corresponding template (again, there are a few use types with more than one template), which indicates the geometric types needed by the predicate representing, very roughly, the ideal meaning of the preposition; and
- tries to find appropriate geometric description functions, as necessary, to coerce the innate geometric type of the object to the type called for by the template, looking at such factors as salience and conventional conceptualizations to determine which functions are most appropriate; and
- returns a completed interpretation consisting of the predicate and objects (modified by the necessary geometric description functions), along with an indication of the which use type was invoked in building it, or returns the information that it can't find any way to build an acceptable interpretation.

3.3.2 Samples of output

Since every use type with appropriate object types is processed, whether or not previous attempts to build an interpretation have been successful, it happens occasionally that more than one interpretation can be made. In general, that is because the locative expression is genuinely ambiguous as between two meanings, as in Example 5 below. The remainder of these examples are fairly straightforward, the program finding one interpretation or none, as appropriate.

Example 1:
(Note: I invented the function `nearest_pt()`, in the absence of any in the book to cover the situation described in *at,3*: the intersection of a linear object and an imaginary line from the speaker (or other reference point) to what would normally be the nearest point on that linear object.)

```
type input - or 'stop.' to end session
|: is there a house at the edge of the park?

preposition and arguments:  at(house,edge(park))

use types for "at":
1  spatial entity at location
2  spatial entity at landmark in highlighted medium
3  physical object at intersection of line and indexically defined crosspath

checking use types for "at" with those objects
object types ok for use type pattern:  3
at(solid_object,line)

no (more) use type patterns compatible with arguments

use types now under consideration:  3
looking for a plausible interpretation
```

```
testing template for use type:  3
coincide(point,point)

possible instantiation of appropriate function types:
coincide(ptapprox(place(house)),nearest_pt(edge(place(park))))

which corresponds to the use type:
3  physical object at intersection of line and indexically defined crosspath
```

Example 2:
(Note: The meaning of "a house on the park" can only be "a house abutting the edge of the park" (*on,4*), since the support relation in *on,1* is not relevant when the reference object is a geographical object. The program correctly returns the same interpretation as this one for "a house on the edge of the park", where the user has supplied the function `edge()` explicitly.)

```
type input - or 'stop.' to end session
|: is there a house on the park?

preposition and arguments:  on(house,park)

use types for "on":
1  spatial entity supported by physical object
2  accident/object as part of a physical object
3  spatial entity located on geographical location
4  physical object contiguous with edge of geographical area

checking use types for "on" with those objects
object types ok for use type pattern:  4
on(physical_object,two_d_object)

no (more) use type patterns compatible with arguments

use types now under consideration:  4
looking for a plausible interpretation

testing template for use type:  4
contiguous(volume,line)

possible instantiation of appropriate function types:
contiguous(place(house),edge(place(park)))

which corresponds to the use type:
4  physical object contiguous with edge of geographical area
```

Example 3:
(Note: The program correctly fails to give any interpretation to this expression; a solid object cannot be said to be included in a line—I did not include in my program the function `stripapprox()`

(approximation to a strip) suggested by Herskovits, which would allow a line to be expanded to an area and would thus permit an interpretation of this expression.)

```
type input - or 'stop.' to end session
|: is there a building in the edge of the park?

preposition and arguments:  in(building,edge(park))

use types for "in":
1  physical object in outline of another, or of a group of objects
2  accident/object part of physical or geometric object
3  spatial entity in area

checking use types for "in" with those objects
object types ok for use type pattern:  1
in(physical_object,physical_object)

object types ok for use type pattern:  3
in(physical_object,two_d_object)

no (more) use type patterns compatible with arguments

use types now under consideration:  1  3
looking for a plausible interpretation

testing template for use type:  1
included(area,area)

testing template for use type:  1
included(volume,volume)

unable to find a plausible interpretation

looking for a plausible interpretation

testing template for use type:  3
included(surface,area)

unable to find a plausible interpretation
```

Example 4:

```
type input - or 'stop.' to end session
|: which car is in the corner of the parking_lot?

preposition and arguments:  in(car,corner(parking_lot))

use types for "in":
```

```
1  physical object in outline of another, or of a group of objects
2  accident/object part of physical or geometric object
3  spatial entity in area

checking use types for "in" with those objects
object types ok for use type pattern:  1
in(physical_object,physical_object)

object types ok for use type pattern:  3
in(physical_object,two_d_object)

no (more) use type patterns compatible with arguments

use types now under consideration:  1  3
looking for a plausible interpretation

testing template for use type:  1
included(area,area)

testing template for use type:  1
included(volume,volume)

unable to find a plausible interpretation

looking for a plausible interpretation

testing template for use type:  3
included(surface,area)

possible instantiation of appropriate function types:
included(base(place(car)),area_from_vertex(corner(place(parking_lot))))

which corresponds to the use type:
3  spatial entity in area
```

Example 5:
(Note: There are two legitimate ways of interpreting the *on* relation in this case: the chimney is contiguous with and supported by the house, and the chimney is an adjunct part of the house. The locative expression "a door on the house" gives the same result, even though the interpretation that the door is resting on the top of the house is relatively unlikely. Further constraints referring to typicality would be needed to rule out that interpretation.)

```
type input - or 'stop.' to end session
|: |: is there a chimney on the house?

preposition and arguments:  on(chimney,house)

use types for "on":
```

```
1  spatial entity supported by physical object
2  accident/object as part of a physical object
3  spatial entity located on geographical location
4  physical object contiguous with edge of geographical area

checking use types for "on" with those objects
object types ok for use type pattern:  1
on(spatial_entity,solid_object)

object types ok for use type pattern:  2
on(object_part,solid_object)

no (more) use type patterns compatible with arguments

use types now under consideration:  1  2
looking for a plausible interpretation

testing template for use type:  1
contiguous(surface,surface)&supported_by(_1458,_1459)

possible instantiation of appropriate function types:
contiguous(base(place(chimney)),overside(place(house))) &
           supported_by(chimney,house)

which corresponds to the use type:
1  spatial entity supported by physical object

looking for a plausible interpretation

testing template for use type:  2
contiguous(geometric_object,surface)

possible instantiation of appropriate function types:
contiguous(place(chimney),surface(place(house)))

which corresponds to the use type:
2  accident/object as part of a physical object
```

Chapter 4

Landscan: what it is, and what it could be

This chapter is by way of being a progress report on Landscan: a brief summary of the work done so far and a presentation of what I see as the next steps toward realizing the goals of the project. This will include proposals for both the near and far term, in greater or lesser degrees of detail, according to the extent of my familiarity with the problems involved.

4.1 What it is

The first-pass implementation of Landscan was intended to demonstrate these capabilities: it was to process the user's query, identify those objects needed to answer the query, and return a visual display, with objects under discussion highlighted. The second pass was to take more complicated questions, using a larger vocabulary and domain model, recognize (that is, distinguish between) more objects, as well as doing some simple analysis of, for example, relations between objects, and return verbal responses, if necessary in addition to the visual display. For the third and later passes, we hoped to handle still more complicated requests, and do more sophisticated analysis, incorporating some rather refined strategies such as asking the user to clarify some part of a query, or moving the cameras and looking at the scene from a different angle or distance to recover information not currently available, as well as making the response behavior less rigid and more cooperative.

In the summer of 1986 a simple demonstration version was written, as an experiment in putting the still rather crude language and vision systems together. That incarnation of Landscan had no real reasoner at all, just a parser and a vision system. The vision component was able to analyse one picture, classifying objects in it according to whether they were at or above ground level (that is, it returned only a binary value for height), and further subdivided each of those groups according to size (i.e., area). The language processor accepted questions about what was visible ("Are there any buildings in the scene?", "What is the distance between X and Y?") and produced a simple Unix command line with a command (`identify, find area/distance, display`) and arguments referring to either generic objects (`sidewalk`) or specific previously identified and numbered objects ([building] `4`). At that point the reply was produced by the vision system itself, verbally in the case of `identify` and `find` questions, visually in the case of `display`, which highlighted the object in question in a partially processed image on a 512 x 512 pixel screen.

The mechanics of command passing were also simple; the object-recognition program, written in C, ran as a sub-process under the Prolog interpreter, and commands were conveyed by writing

to and reading from files.

Developments since that time include Anderson's recent work, mentioned in section 1.1.3, and my own, outlined in Chapter 3. In addition, a good deal of reading, thinking, and talking about the project has yielded the comments that follow, outlining my view of what should happen next.

4.2 What's next?

There are quite a number of things that need to be developed within Landscan, and one necessary facet of the development of any system is adherence to good software engineering principles. Specifically, modularity and top-down, structured design. Early work on the system was largely experimental, to test the limits of what could be done at both the language and the vision ends with current resources and to determine what new resources would be required. What was learned from this period is that the next phase of the project must be carefully planned as a single, unified system, and designed with that unity in mind. Further, the capabilities of the system as a whole, with respect to the project's original charter, that is, to use the input to guide the work of the rest of the system, in particular the vision component, depend crucially on a contemplation at once detailed and global of what should actually occur in each part of the system as a request for information is being processed.

4.2.1 Modules and modularity

As I mentioned in the introduction (Chapter 1), the overall structure of Landscan was seen as tripartite: it would contain a vision system,[1] a language processor, and a reasoner to act as the interface between them. The working of the reasoner, which is conceptually prior to any interaction between the modules, has received the least attention to date. In particular, input and output for each module must be specified, so that design within the modules may proceed from those specifications, in addition to specifications of the behavior desired of each module.

Obviously, both of the "end" modules, the vision and language systems, have known where their raw materials were coming from: the data returned by the cameras in the one case, and the user's input query in the other. But what each would produce, and in what form, must now be considered, with respect to the requirements of the system in general and the reasoner in particular.

4.2.2 Designing the reasoner

The first step is to design the reasoner, in some detail. In particular, the design must specify the languages to be used in every interaction, as well as the resources available at the various points in the processing sequence. As I see it, development of the reasoner/interface will touch on a variety of issues, and involve a number of critical decisions.[2]

[1] The vision system itself is also a multi-layered design, from the lowest-level processing up to object recognition, which requires considerable intelligence as well as knowledge about objects and the way they look. In fact, there is a good deal of reasoning going on within the vision component, but my references to it will be as a single part of the Landscan project; I think it will be necessary for the reasoner, at least, to see it that way.

[2] I am avoiding the use of the word *interface* alone to refer to this central reasoning module, though heretofore I have thought of it and referred to it as an interface between the language and vision systems, because I'd like to keep that word to refer to the protocols for the exchange of information between modules, to be able to speak, for example, of the communication between two modules as taking place through a specified interface. I will thus refer to it henceforth only as the *reasoner*.

Input

It should receive from the language processor a fully disambiguated representation of the input query, which must (minimally) contain information as to what kind of question is being asked (yes/no, request for identification or location of an object, request for some geometric datum such as distance or size), which objects, whether already identified or not, are being referred to, and if relevant a spatial relation between those objects.

The language that will be used to represent this query should be elaborated with attention to the degree of semantic sophistication needed; it is my impression that something like Woods' MRL [26] contains more information about quantification than we need here, but I may be wrong. I have been thinking along the lines of a predicate with arguments representing the question type as outlined above (is there, how many, where is, yes/no, etc.) and the object(s) in question, which will need to be specified using fairly fine-grained descriptors. It may turn out that that sort of representation is too narrow; to my mind it would cover virtually all of what Landscan is expected to be able to reason about.[3] In any case, the point is that the needs of the reasoner should determine the shape of the language processor's output.

A second input will come in the form of information being returned by the vision system; a language for that process, too, will have to be specified to enable the further reasoning that must be performed before an answer is passed on for generation of output to the user.

Output

The reasoner will output in two directions: commands to the vision system, and some representation of the answer to the language processor.[4] Commands to the vision system should be on the highest level of abstraction the vision system can deal with[5] and of course the interface between these modules should be as simple as possible.

"Mental map"

The reasoner will have to build and maintain a (necessarily incomplete) picture of the scene as the conversation progresses. If we specify that *all* relevant information generated by the vision system in response to any request be returned to the reasoner, the latter will end by having, in theory at least, a good deal more knowledge about the scene than the user has asked for. For example, if the user asks which is the largest building in the scene, the sizes of all buildings will have to be computed. It makes sense for the reasoner to store all of them, in case any question regarding size comes up later. This is in line with the notion that calls to the vision system should be minimized; certainly data of this kind should never have to be generated more than once.

The language for this representation will also have to be specified; I envisage it as the same as or highly compatible with the knowledge representation language chosen for the domain as a whole.

World knowledge

This may be the most difficult part of the system to design. Knowledge about the domain, in a great deal of detail, is required here, and the most important question to be answered in designing

[3] A semantic representation for input questions will be discussed at greater length in section 4.2.4.

[4] Actually it is unlikely that the same part of the language processor would deal with both understanding and generating; I ignored the generation question in my work, since I saw the output as being formulated using templates, which would imply a very simple answer-production module.

[5] See footnote 1, this chapter.

Landscan is that of which knowledge belongs where, and in what form. Among the things the reasoner may have to know about is the disparity between the visible scene and some abstract "reality" that must be presumed to underlie the picture, starting with the perhaps difficult-to-capture fact that the "world" has three dimensions and the internal representation of the image only two-and-a-half (the vision system will be responsible for the prior translation from the 2-D of each of the initial images to $2\frac{1}{2}$-D). It will have to be aware of the possibility of partial or total occlusion, to be capable of knowing, for example, that some objects, such as buildings, have interior spaces, with shape characteristics that cannot be ascertained from looking at their outsides.

I have come to feel that some knowledge, particularly various kinds of knowledge about objects, will have to be duplicated, that is, reside in more than one part of the system.[6] For example, the definitions of objects the vision system uses to isolate and identify them, particularly knowledge about the context in which a given object can occur, will also have to be available to the reasoner, if only so that its "mental map" will be coherent. I will have more to say about representing knowledge in section 4.2.5.

System knowledge

It will be necessary to specify a precise "instruction set" of commands the vision component can handle; this will constitute the reasoner's knowledge of what the vision system is capable of. Similarly, the language used to signal output to be generated to the user will constitute its knowledge of what the language generator does. But, to repeat, since the reasoner is the controller for all internal processes (i.e., all processing that takes place after the initial interpretation of the input), all the protocols for sending requests to other modules and receiving replies must be designed around its needs.

Reasoning

I have been envisaging the actual reasoning, that is, determining what information is needed, what, if any, is already available (either in the pre-existing knowledge base or in the current "mental map" of the scene, or derivable from the information contained in one of those), and what must be sought from the vision system as a rule-based operation. In this view, the reasoner is rather like an expert system whose expertise is in contemplating scenes and answering questions about them.

If the user asks, for example, "Is there an X next to Y?", where Y is a singular definite reference, a number of steps must be taken to find the answer; I will elaborate one possible sequence of such steps, but in fact there are decisions to be made about the design of the system at almost every point in the list that follows.

1. see that Y is indeed already identified and where it is (using some system of coordinates, presumably, possibly indicating corners or a center point), by referring to the "mental map" of the scene, wherever it is being kept; then, assuming Y is already known:

2. see whether any X has already been recognized, and if so:

3. test for the relation *next to* with respect to that X and Y; alternatively, one could check symbolically only in the vicinity of Y for known Xs, and test whether any of them can be said to be *next to* Y (the use of this strategy is of course specific to certain prepositions); if the relation doesn't hold for any such X, or if no X has yet been recognized:

[6] Unless, of course, a single database/domain model is available for access by any module in the system.

4. ask the vision system to find an X in the scene *or* search only the area within a certain radius of Y looking for an X, that is, let the input question `is_there(X),next_to(X,Y)` (or whatever representation is chosen) constrain the area under consideration during object recognition—the choice between these two strategies will depend on whether restricting the area of the image to be processed in this way is actually possible and actually would limit the amount of work the vision system does;[7] a consideration relevant to that decision (as well as to the overall specifications for the system) is how we want the system to answer if there is an X we would call *near* Y, but not *next to* it, according to our definitions of those relations, or if we find an X next to some other object of the same type as Y, and so forth, in other words, how cooperative should responses be and how should they be cooperative?

5. test whatever X is found to see if it is next to Y—ideally, we would like a definition for the relation *next to* that captures the kinds of normal situation type and pragmatic factors (typicality, tolerance) outlined by Herskovits; for example, two buildings can be said to be next to one another if they are relatively close (relative to their size and to the nearness of other buildings, among other things), even if there is some much smaller object such as a tree between them, but two trees the same relative distance apart but with another tree between them cannot;

6. depending on the decisions made with respect to cooperativeness and what to do when a presupposition such as the existence of the reference object fails, send to the output part of the language module some representation for *yes*, or *no*, or *no, but there is an X* near *Y*, or whatever is appropriate.

4.2.3 What we should expect of the vision system

Geometric description functions

As was briefly noted in the previous chapter, Herskovits has contributed not just a linguistic approach to locative expressions, but a whole account of spatial cognition that seems complete, sound, and potentially very useful in the design of a system occupied with spatial relations. Her paradigm, and in particular the notion of geometric description functions helping to define the location of an object, give us a reasonable way of approaching a language for the precise kinds of information we might want the vision system to be able to return with respect to objects in the scene. That is, the representations she proposes are a useful construct for designating the idealizations, parts of objects, and so forth, we would like the system to be thinking about in calculating the answers to questions about spatial relations between objects; I am proposing that some form of her predicate representations for locative expressions is exactly what we need in reasoning about objects in space as well as talking about them. The vision system will therefore have to be able to compute functions like the ones in her paradigm.

As an example, take the function `ptapprox()` (approximation to a point). We saw that, for the purposes of the meaning of *at*, objects had to be seen as points, which requires, among other things, that the vision system be capable of finding for an object under consideration a point in the $2\frac{1}{2}$-D representation of the scene (or in the imaginary 3-D scene it corresponds to) that can be used in reasoning about its location.[8] Then the information that `ptapprox(place(X))` is `(A,B,C)`

[7] In any event, the fact that it is an X that is being sought may be able to help the object recognizer decide what it is looking at. More on this in section 4.2.3.

[8] It might be necessary to define the function `ptapprox()` differently with respect to different prepositions, or according to the geometric type of the object we want to be approximated, in which case additional constraining

and `ptapprox(place(Y))` is `(D,E,F)` (where `A`–`F` are integer coordinates in the representation of the scene) can be used by the reasoner in determining, according to defined tolerances based on typicality, etc., whether those two points are close enough together for it to conclude that X is *at* Y.

This means that the vision system must have procedures for finding the approximations to points, lines, or surfaces, as well as edges, main axes, and so on that contribute to the meanings of the locations of objects. Given a meaningful image segmentation, some of these functions are easy to define (edge, outline, main axis), others somewhat harder (approximation to a line, area associated with a vertex); still others, such as completed enclosure (an imaginary enclosure that is a projection from a few points; in "the houses around the lake", the houses can be thought of as implying an enclosure that surrounds the lake) and other idealizations involving projections of that sort, would be very difficult indeed.[9] Of course, an understanding of what exactly the functions will be used for, and of how testing of relations based on them will be effected, will in turn affect how they are defined. But one of the next steps for the vision system is a careful look at which of these functions can be implemented, what knowledge is required to formulate definitions for them, and how much scene understanding is needed before they can be calculated.

Heights

The vision system is already capable of computing distances and areas of objects; in addition we will need greater accuracy in height calculation; currently height is returned as a boolean indicating whether an object is at or above ground level. This has largely to do with photographic equipment, and improving the accuracy of these calculations is a current research goal of the Grasp (robotics) Laboratory here. Height distinctions, which come down to the ability to compute the length of the Z axis when we are looking directly down at the scene, both as between objects and within an object (to calculate, for example, the slope of the roof of a building, or in fact the slope, if any, of the ground), will be required if the system is to be capable of dealing with the range of questions we hope at some point to be able to ask.

Object recognition

The disambiguation of objects by a recognition program relies heavily on a cleanly modeled domain; object definitions, however they are represented, must be formulated to provide the most complete and discrete coverage possible of the range of shapes encountered in the scene. That is, there should ideally be little or no overlap between definitions; we would like each object to be uniquely delineated. To that extent, Anderson's discrimination tree approach [1], after Mulder [18], is intuitively appealing, and although I have some reservations about how easily it can be extended to provide the much finer distinctions ultimately needed in Landscan, I am inclined to want to take that approach as far as it can be taken before turning to a more complex recognition system, such as a production system, for example.

One element that will eventually have to be incorporated in object definitions is the notion of context. Part of the definition of an object should be the context in which it typically occurs, possibly including proscriptions, situations in which it may not occur. Thus, for example, we would want the recognizer to reject the identification of a sidewalk in the middle of a street, even if its

information will have to be passed to the vision system, along with the identity of the object in question, when a point approximation is needed. Obviously the same will be true in various ways for all these functions.

[9] Helen Anderson, personal communication.

parameters fit the definition for *sidewalk*, just because the context is too unlikely. The presence of information about context will make object definitions both more precise and more reliable.

Guiding image processing

One very important facet of the Landscan project is the exploration of the question of how image processing might be guided by the input. A point that now demands attention is how the vision system can make use of information contained in the user's query to help it to understand the image. The most important level at which that will be useful is in the segmentation process, which is the phase immediately following edge detection, and the first in which the system is trying to find meaning in the image. At this point, knowledge of context, that is, of what it should expect to find, can be enormously helpful in guiding the decisions the vision component will need to make concerning which of the many edges that have been detected are likely to be meaningful, and how. Anderson has already begun to address this problem.

Constraining image processing

So far, the possibility of restricting the amount of image processing necessary has gone largely unexplored, but since another of the original goals of the project was to look at whether and how far that might be done, it would be well to enter the next phase of design of the vision component with that question also in mind.

It has been suggested that certain phases of the image processing are not amenable to an approach that would divide the image and consider only part of it. It has also been proposed that some kinds of processing *could* be restricted to parts (quadrants, for example, or horizontal or vertical strips) of the image, but how far that idea might be carried has not, I believe, been determined. Strategies such as those proposed in Chapter 1 can be used to limit the number of requests made to the vision system, but we would like as well to be able to narrow the area being searched for a particular object, for example, if that could lighten the image-processing load.

4.2.4 What we should expect of the language processor

Semantics

At the core of the design of the language processor is the question of what the semantic representation of the input sentence should be. One conventional answer to this question is some kind of predicate calculus representation, such as MRL [26] or a logic grammar-produced representation, e.g., Pereira and Warren's [19] or Dahl and McCord's [8]. But these representations, as they are usually formulated, do not respond precisely enough to the needs of this system, which requires specifically notions of how to refer to objects, how to refer to locations, which will normally be expressed as relations between objects (as is conventional in interactions between people), and how to identify what is being asked about the objects and locations.

An approach I would want to pursue is to think in terms of a semantics specifically for questions, one that is designed to capture the particular meaning of interrogative sentences, as the grammar captures their particular syntax. What I have in mind is to represent explicitly the given/new distinction that I see as implicit in any question, the sense of "*this* is what I know, *that* is what I want to know about it". If we ask "How tall is X?", the fact of X's existence is given; its height is what is being queried. This reflects exactly the analysis in Clark and Haviland [6] of some corresponding declarative sentence like "X is Y metres tall", in which the fact that X is some number of metres tall is given and the information that the number is Y is new.

One representation that seems plausible for implementing this approach is presented in McCord's paper on focalizers [15], in which the scoping of quantifiers is treated by giving to the quantifier two arguments, representing a pair he calls Base/Focus, the base representing, to all intents and purposes, what is known or presupposed, and the focus what is being said about it. It seems a short step to a similar pair Base/Query, corresponding to the given/new distinction in questions outlined above. Indeed, what I am proposing is simply that we make that distinction explicit in the representation.

In addition, a richer-than-usual vocabulary is needed to represent objects and their locations in this domain. A representation very much like the interpretation of a locative expression arrived at by my program would, I think, provide an adequate degree of detail.

Thus I envisage (roughly) a representation of a user's query like "Is there a tree at the edge of the park?" as composed of a top-level predicate that constitutes the question type, in this case `is_there`, with arguments corresponding to the base and query, that is:[10]

```
indef(tree)&def(edge(park)), coincide(ptapprox(tree),nearest_pt(edge(park)))
```

where the predicates `indef(X)` and `def(X)` simply mean X has been referred to using an indefinite or a definite reference; these correspond loosely to existential and universal quantification over the noun phrases. The fact of `is_there`'s being a predicate is actually irrelevant; this could just as easily look like a command line, as in previous versions of the system. It is important, though, that the locations of *tree* and *edge of the park* be fully specified by the necessary geometric description functions, to facilitate testing of the relation in question, which I see as critical to the system's ability to answer questions about location accurately.

Parsing the whole question

Obviously, the program for interpreting locative expressions described in Chapter 3 is still an inadequate language processor for the system as a whole, because it does not process the entire question, and thus does not produce output that can be reasoned about. I see that program as a subroutine contributing the fully interpreted locative expression to the final representation of the query sent to the reasoner. Completing the query representation process along the lines described above seems to be the next step for the language system. Clearly, more thought will have to be given to exactly how the parser will work; I am thinking in terms of a grammar that is also geared specifically toward interrogative sentences, focusing on how the various parts of a question contribute to the meaning, that is, assuming that some part of the sentence represents given information and another part the object of the question.

4.2.5 Knowledge representation: eternal bugaboo

It has been noted, and I hope amply shown, that the representation of knowledge is a problem for every part of a system like this one, which suggests that a unified approach to the question, a single domain model/knowledge base serving all of the other components, might be worth trying.

The notion of a central, universally accessible knowledge base is appealing on several counts: it ensures that the knowledge a module uses in any reasoning it does is consistent with what every other part of the system "knows"; it helps to increase the modularity of the system as a whole; and of course it obviates the duplication of data. On the other hand, if the knowledge base is to contain information about objects to be used by both the reasoner and the vision system, for example, it

[10] I am proposing dropping the use of the predicate `place()` for all objects, since I feel we can assume in any locative relation that it is always some aspect of the overall place of the object we are interested in.

must be in a form that is usable by both for their respective tasks. Finding a single representation that is sufficiently versatile may not be easy.

For instance, I have been seeing some kinds of knowledge, such as that used by the reasoner in considering how to gather the necessary data to form an answer to a query, as directly expressed in rules, while others, such as the description of the domain of objects, seem amenable to placement in a hierarchy of concepts further linked by roles and restrictions, à la KL-ONE. These two styles should not in principle be difficult to marry in a single system, but whether they will suffice for managing the knowledge we need will have to be seen.

Meta-knowledge

I have been assuming that "meta-knowledge" about the fact that we are in a two-and-a-half-dimensional world that represents a three-dimensional world would be represented in the vision system's knowledge in a system with separate knowledge bases. With a unified KB, of course, the question is simply which part of the system will use that knowledge; I suggest that it be primarily, if not exclusively, the vision system that does so. That is, by the time information reaches the reasoner, it should reflect the three-space meaning the reasoner works with. Similarly, requests to the vision system for information would be expressed in terms of three-space. On this model, all "translating" between the two codifications of spatial reality would take place in the vision system.

Another useful piece of meta-knowledge is the fact that the conversation is taking place with a user, whose perspective on the scene must be taken into account, for example, whenever a projective preposition is used. This would be reflected in the definitions of those prepositions, which should as a default assume that *to the left of* means to the viewer's left in the image plane, and that *behind* qualifies something partly or wholly occluded (the *encounter situation* interpretations; the user's having defined a "front" of an object earlier in the exchange would license a *coincidence situation* interpretation).[11]

Knowledge of the existence of the user, and of the fact that the interaction is taking place outside of the "reality" under discussion, would permit the use of (first- and second-person) indexical pronouns, as well as making concepts like "in the scene" meaningful.

4.3 Long-range goals

Given the current resources of the system, I believe the following ideas are still a long way from being realizable; the structures needed to support them, in particular the reasoning activity and the representation of knowledge, must be designed and realized first.

4.3.1 Mixed-initiative interactions

One of the possible ways of dealing with problems encountered in processing an input question is to ask the user to clarify an intention, or define a term, or perhaps reformulate a question. This notion was conceived with an eye to making the interaction between the user and system potentially a learning experience for the system; new terms (such as *old*, as in the example in section 1.1.1), could be introduced into the conversation, defined, and incorporated into the system's knowledge. Or a definite reference failure, for example, could be queried and perhaps corrected ("There is no

[11] The *encounter situation* is that in which the meaning of the preposition is assigned with reference to the speaker's own orientation; the *coincidence situation* occurs when the speaker's perspective and that of the object imaginatively coincide, that is, the speaker refers to the *object's* left or front. These notions are explained fully in Chapter 10 of Herskovits' book [12].

statue in that field; do you mean the tree?") so that processing of the question could resume, or start again with different objects as arguments.

I see two critical issues coming up here:

- In the first place, the possibility of, in effect, discussing the user's input with her/him presupposes that the gap in the reasoner's knowledge that caused the processing problem can be identified.
- In the second, any new knowledge thus acquired must be incorporated into existing knowledge bases, and all the problems of maintaining consistency in a dynamic domain model are promptly encountered.

The idea seems most straightforward in the case of an unknown word or expression; a representation for the input will contain a variable where the reasoner is expecting a constant, or perhaps a reference to a concept that is not contained (or simply not known by that name) in the knowledge base. In that situation, it seems reasonable to output a request for a definition but, as was noted in the example with *old*, this also requires that the system know what kind of definition would be useful—in this case a description of the visual characteristics of an object that can be qualified by the word *old*, and specifically what an *old building* looks like, since the word was used in that context.

The sources of other failures to "understand" the input request (beyond the level of a failed parse) will not, I think, be so easy to identify. If there are several possible places for the error to have occurred (incorrect definite reference, incorrect description of the location of an object, location with respect to a non-existent reference object, etc.), it's not at all clear how the decision about what clarifying information to ask for might be made.

Assuming the user responds in a constructive way, we would like at this point to update (1) the dictionary, to include the new adjective, (2) the domain model, to add the concepts *old* and *old building*, properly placed in the structure (that is, the latter subsumed by the concepts *building* and *old*, still assuming a KL-ONE sort of structure), and (3) the set of rules that define objects in terms of their visual characteristics, to add the definition just supplied by the user. The domain model, at least, must then be checked for consistency with respect to existing concepts and roles; probably some coordination of the new visual definition with existing rules will also be needed.

4.3.2 Look-again strategy

Another interesting and rather visionary idea was to have the system know when it would be useful to look at the scene, or some particular object in it, from a different angle or distance, in order to ascertain some datum not available from the current perspective. This would normally be due to partial occlusion of one object by another from a given angle, or to insufficient resolution due to distance.

One major problem in implementing this, from the point of view of system design, is how to represent the "meanings" of occlusion and insufficient resolution, that is, for example, what it means for an object to lie *behind* another, and thus how such a change in perspective might be expected to yield useful information. In other words, I see this as a problem in reasoning about the visual situation. A second problem is that of calculating how and where the cameras need to be moved. Still another is managing the addition of the newly-acquired visual knowledge to the system after the cameras have been moved, that is, how to merge it with the existing knowledge of the scene.

A related question is whether it is the reasoner that should have access to the necessary information about current camera position, in addition to being able to calculate where to move

to. Should this be a part of its system knowledge, or would it make more sense for another agent entirely to reason about both the mechanics of moving the camera and the rationale for doing so? Work is being done in the robotics field that may be helpful in addressing these questions.

4.3.3 New technologies

Parallel processing

In the introduction to Chapter 3, I posited that many of the rules for finding an appropriate interpretation of a locative expression were essentially non-procedural and functionally independent, and thus at least some of the tests involved could take place in parallel. Specifically, the testing of the several use types for a given preposition (section 3.1.3) could certainly be done simultaneously; every use type must be similarly examined, and none of the testing depends in any way on the results of testing any of the other use types. In general, though, the processes preceding and following that "template filling" can only be sequential, since dependencies do exist between most of the other routines.

Likewise, in reasoning about the input and how to respond, it might make sense to call for a number of tests on what is known about the objects under consideration at the same time: if several independently derivable data were needed from the vision component, they could all be asked for in a single request and calculated simultaneously; again the question of dependencies will have to be kept in mind when the processes are designed. I make the assumption, though, that the reasoner would have to be conceived with the possibility of parallel processing already in view, that is, thought would have to be given at the outset to shaping the reasoner's work in such a way as to exploit the benefits of parallelism.

Some visual processing is amenable to being performed in parallel; convolutions can be done on an entire image at once, again because within the application of one convolution mask to all the pixels in an image there are no dependencies between results being produced in that one application. Stereo matching is another candidate for multi-processing.[12] A good deal of work has been done on utilizing parallelism in image processing, in particular using special-purpose architectures, but applying it in this case would require an early design decision and implementation-specific software in addition to the necessary hardware.

Connectionism

A certain number of the processes involved in Landscan seem to lend themselves to a treatment involving weighting likelihoods of occurrence, preferring or ranking interpretations with respect to some notion of typicality or reliability. If such a treatment were to be implemented, a connectionist network approach, involving dynamic evaluation and adjustment of the weights, might be effective.

Examples of points in the various parts of the system where this could be useful are:

- in language processing: the judgments to be made regarding relative appropriateness of particular interpretations of a locative expression (it may likewise be possible to weight other aspects of the interpretation of the input according to the likelihood that a given meaning is intended);
- in object recognition: the relative probability of an object's being one thing rather than another, especially with regard to the question of plausibility in a particular context, as was discussed in section 4.2.3;

[12] One visual processing problem that is *not* a good candidate for multi-processing is segmentation, because the formation of coherent units is dependent on global context.

- in reasoning: it seems sensible to think of the result sides of rules defining spatial relations as having weights, or some way of ranking them, since typicality and likelihood are also pertinent in determining which spatial relations might hold between two or more objects.

The value of a ranking or weighting approach is especially clear-cut in the object recognition case; ranges of acceptable values for the various parameters of an object could be included in their definitions, and a categorical determination that an object is an X would mean that the parameters identified fell within those ranges. An object like the clothespin sculpture (a three-story-tall clothespin by Claes Oldenburg in downtown Philadelphia), might then be seen as unclassifiable: in such a case, where the standard tolerances would be inadequate to cope with the degree of deviance from the "typical" parameters, we might want the system to be able to say of the object, with some probability rating, "It could be a building, but its height-to-base-area ratio is anomalous." It would presumably be possible to have another reliability reading for the interpretation that the object was a sculpture, or a pylon for high-tension wires; all of these could then be readjusted in light of information subsequently obtained about the scene.

4.4 Conclusions

The main lesson learned from early phases of Landscan work is that it is essential that the system be designed from the top. The details of how much and what kinds of integration are required need to be settled quickly, along with clear specifications for communication between modules.

A related need is for an in-depth examination of the knowledge representation problem: what knowledge is needed, how it will be used, how it is to be represented. A study might be undertaken of the feasibility of a single, unified knowledge base for the whole system. If it is a possibility, that approach would obviate the problems of duplication of data and consistency between different representations; it therefore merits consideration before any commitments to module-specific ways of storing knowledge are made.

A third question that is crucial to the development of the project is that of how an input query really might influence the system's methodology for responding to it. This will entail in the first instance an investigation of how the user's expectations can help the vision component to isolate and identify objects, but it may be that the behavior of the reasoner in determining what information is needed to formulate a response could also be guided by, for example, the presuppositions and entailments of the question. And that in turn will depend on the language processor's ability to accurately represent what the user wants to know.

Understanding the input, then, at a level of sophistication and subtlety that captures such things as the intention implicit in a definite reference, or the user's apprehension of the location of one object with respect to another (implied by the preposition chosen to represent that relation), is the support on which every other function depends. For that reason, it is essential that a means of capturing these and other aspects of the input be found.

One contribution to attaining the necessary degree of refinement with respect to objects and their locations is a representation for locatives like the one Herskovits proposes: the objects themselves are viewed in terms of how they are spatially related; this is exactly the information we need about them in a context in which geometric relations between objects is to all intents and purposes the entire topic of conversation. The understanding we have gained from her work of how idealizations, approximations, synecdoche, projections, and so on contribute to the meaning of location for objects is vital, and I feel the paradigm ought to be not only used for interpreting the input but also incorporated in the representation of the system's knowledge of objects and spatial relations, and thereby inform its reasoning as well.

I have remarked that the idea of Landscan is an ambitious and demanding one, seeking as it does to exploit the user's own questions in novel ways to help in answering them, and addressing itself to many of the quintessential difficulties in artificial intelligence, but the work done so far has yielded important information on how the project should be pursued from here, and gives reason for optimism. We have a clearer and more realistic view than heretofore of what is needed, and are now in a better position to proceed with developing the system.

4.5 Acknowledgments

My deepest appreciation to the Landscan group: Aravind Joshi, for invaluable guidance and support; Bonnie Webber, for many helpful and clarifying remarks, commentaries, and criticisms; Helen Anderson, for her valiant efforts to increase my understanding of machine vision, and for helping me see important similarities, and some equally important differences, between understanding language and understanding images; Ruzena Bajcsy, for believing in me. And to Brant Cheikes, for his unstinting support in this and every endeavor.

Bibliography

[1] Helen L. Anderson. *Edge detection for object recognition in aerial photographs.* Technical Report MS-CIS-87-14, Grasp Lab 96, Department of Computer and Information Science, University of Pennsylvania, January 1987.

[2] Helen L. Anderson, Ruzena Bajcsy, and Max Mintz. *A modular feedback system for image segmentation.* Grasp Lab Report (forthcoming), Department of Computer and Information Science, University of Pennsylvania, 1987.

[3] R. Bajcsy and A.K. Joshi. The problem of naming shapes: vision-language interface. In David L. Waltz, editor, *Theoretical Issues in Natural Language Processing-2*, pages 157–161, University of Illinois, Urbana-Champaign, IL, 1978.

[4] Ruzena Bajcsy, Aravind Joshi, Eric Krotkov, and Amy Zwarico. LandScan: a natural language and computer vision system for analyzing aerial images. In *IJCAI-85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 919–921, Los Angeles, CA, August 1985.

[5] R.J. Brachman. *A structural paradigm for representing knowledge.* Research Report 3605, Bolt Beranek and Newman, Cambridge, MA, 1978.

[6] Herbert H. Clark and Susan E. Haviland. Comprehension and the given-new contract. In R. Freedle, editor, *Discourse Production and Comprehension*, pages 1–70, Lawrence Erlbaum, 1977.

[7] Veronica Dahl. Translating Spanish into logic through logic. *American Journal of Computational Linguistics*, 7:149–164, 1981.

[8] Veronica Dahl and Michael C. McCord. Treating coordination in logic grammars. *American Journal of Computational Linguistics*, 9:69–91, 1983.

[9] H. Paul Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and semantics, vol. III*, Academic Press, New York, 1975.

[10] Patrick J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert systems in the microelectronic age*, Edinburgh University Press, Edinburgh, 1979.

[11] Martin Herman and Takeo Kanade. Incremental reconstruction of 3D scenes from multiple, complex images. *Artificial Intelligence*, 30(3):289–341, December 1986.

[12] Annette Herskovits. *Language and spatial cognition: an interdisciplinary study of the prepositions in English.* Cambridge University Press, New York, 1986.

[13] Annette Herskovits. Semantics and pragmatics of locative expressions. *Cognitive Science*, 9:341–378, 1985.

[14] Aravind K. Joshi. Mutual beliefs in question-answer systems. In N. Smith, editor, *Mutual Belief*, Chapter 4, Academic Press, New York, 1982.

[15] Michael C. McCord. Focalizers, the scoping problem, and semantic interpretation rules in logic grammars. In: *Logic Programming and its Applications*, ed. M. Van Caneghem and D.H.D. Warren, 1985.

[16] Michael C. McCord. Modular logic grammars. 1985. Unpublished working paper.

[17] C. S. Mellish. *Computer interpretation of natural language descriptions.* Ellis Horwood Ltd., Chichester, 1985.

[18] Jan A. Mulder. *Using Discrimination Graphs to Represent Visual Knowledge.* PhD thesis, University of British Columbia, 1985.

[19] Fernando C.N. Pereira and David H.D. Warren. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278, 1980.

[20] Leonard Talmy. Figure and ground in complex sentences. In J. H. Greenberg et al., editor, *Universals of human language*, Stanford University Press, Stanford, CA, 1978.

[21] Leonard Talmy. How language structures space. In Herbert Pick and Linda Acredolo, editors, *Spatial Orientation: Theory, Research, and Application*, Plenum Press, 1983.

[22] Leonard Talmy. The relation of grammar to cognition—a synopsis. In David L. Waltz, editor, *Theoretical Issues in Natural Language Processing-2*, pages 14–24, University of Illinois, Urbana-Champaign, IL, 1978.

[23] David L. Waltz. Generating and understanding scene descriptions. In A. Joshi, I. Sag, and B. Webber, editors, *Elements of discourse understanding*, Cambridge University Press, Cambridge, England, 1980.

[24] David L. Waltz. On the interdependence of language and perception. In David L. Waltz, editor, *Theoretical Issues in Natural Language Processing-2*, pages 149–156, University of Illinois, Urbana-Champaign, IL, 1978.

[25] David L. Waltz and Louise Boggess. Visual analog representations for natural language understanding. Working paper presented at IJCAI-79, Tokyo, August 1979.

[26] William A. Woods. *Semantics and Quantification in Natural Language Question Answering*, pages 2–87. Volume 17, Academic Press, New York, 1978.

[27] Amy E. Zwarico. *The recognition and representation of 3D images for a natural language driven scene analyzer.* Technical Report MS-CIS-84-29, Department of Computer and Information Science, University of Pennsylvania, 1984.