Cover Design: On the cover is a schematic of the principles underlying a model-based system for the understanding of 3-D images. In this particular system, data is presented as a 3-D image (a portion of which is shown, pseudo-colored, on top left) comprised of a set of Computerized Axial Tomograms (CAT scans) of the human abdomen. Together with a knowledge-based geometric model of the abdominal anatomy (based on Generalized Cylinders: top right and bottom left), it is fed to a program that provides a model-instance of the recovered actual organs' shape (right: two kidneys and a portion of the spinal column). All the software and techniques for the system, and its graphics have been developed entirely in the Department's laboratories, and comprise a major portion of an upcoming PhD thesis by Uri Shani.
<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Experiences with the Generalized Hough Transform,</td>
<td>K. R. Sloan and D. H. Ballard</td>
</tr>
<tr>
<td>9</td>
<td>The Rochester Natural Language Understanding Project,</td>
<td>J. F. Allen</td>
</tr>
<tr>
<td>17</td>
<td>An Information-Theoretic Approach to Time Bounds for On-Line Computation,</td>
<td>W. J. Paul, J. I. Seiferas, and J. Simon</td>
</tr>
<tr>
<td>28</td>
<td>Publications</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Seminars</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Grant Support</td>
<td></td>
</tr>
</tbody>
</table>
EXPERIENCE WITH THE GENERALIZED HOUGH TRANSFORM

K. R. Sloan, Jr. and D. H. Ballard

1. Introduction

Shape is an important attribute of two-dimensional figures. In simple figure-ground binary images, the shape of the boundary of the figure is often the only interesting feature. We take "shape" to be a property of the entire figure, i.e., it is a global property.

Evidence about the shape of a figure is found at the boundary between figure and ground. Such evidence can be generated by the application of local edge-element detectors. An edge-element detector typically reports on the presence of an edge-element in a small window of an image, and on the orientation of that edge-element. Finding shapes in the image involves combining many pieces of local evidence into a global judgment.

The Hough Transform is a method for detecting curves by exploiting the duality between points on a curve and parameters of that curve. The initial work showed how to detect both analytic curves [Hough, 1962; Duda and Hart, 1972] and non-analytic curves [Merlin and Farber, 1975], in the case of binary edge images. This work was generalized to the detection of some analytic curves in grey level images, specifically lines [O'Gorman and Clowes, 1973], circles [Kimme et al., 1975], and parabolas [Wechsler and Sklansky, 1975].

Recently, the Hough technique has been extended to the detection of arbitrary non-analytic shapes in grey level images [Ballard, 1979]. Given an arbitrary shape, S, this generalized Hough technique provides a mapping from the orientation of an edge-element to the set of instances of S (as modified by location, rotation, and uniform scaling) which could have given rise to that edge-element. This mapping allows all local evidence for a particular instance of S to contribute to global decisions about the figure.

This shape detection scheme has been implemented and tested on a variety of artificial images and has found application in the analysis of real aerial images. Experience to date indicates that the technique is robust. Also, with appropriate "focus of attention" mechanisms, which are present in our implementation, the method is also efficient. However, the reliable determination of edge-element orientation is crucial to the success of this method.

2. Hough Techniques

All Hough techniques for shape detection consist of the following basic elements:

a) a local edge-element detector, E,

b) an n-dimensional parameter space, P, quantized and represented by an n-dimensional Accumulator Array, AA,

c) a mapping, M, from the information provided by E into P (and thus AA),

d) a voting rule, V, specifying how a particular edge-element affects the values of AA,

e) a Detection rule, D, specifying the conditions under which a particular shape has been detected.

Given these basic elements, shapes are found by the following procedure:

a) zero AA,

b) apply E everywhere in the image,

c) for each edge-element found, apply M to locate cells in AA. Then apply V to modify the contents of these cells. (i.e., vote for all possible "causes" of this edge-element),

d) finally, apply D to AA (choose the most popular shape).

Clearly, application of this technique depends on the ability to parameterize the shapes of interest, and the derivation of the mapping M from edge-element information to possible shape parameters.

Lines

The original Hough transform capitalized on the observation that straight lines can be completely specified by two parameters (e.g., an orientation [theta], and a distance from the origin, s). What is more, the mapping, from a particular edge-element position to the set of straight lines it might be a part of, is easy to compute [Hough, 1962; Duda and Hart, 1972]. The idea is that an actual line in the image will give rise to many local edge-elements, all of which will "vote" for that line. Individual edge-elements will also vote for other lines, but the "correct" line will receive the most votes.

If the edge-element operator, E, provides directional information, then each edge-element maps to a unique line. Edge elements which line up vote for "their" line, and the line with the most visible edge-elements gets the most votes. Note that it is not necessary for the edge-elements to be connected (or even be near each other) in order that their votes reinforce one another - they must simply be collinear.
Circles

The description of circular figures in an image requires three parameters: x, y, s. The location of the center of the circle is given by (x,y) and the radius is given by the scale parameter, s. Each edge-element in the image is evidence for a set of (x,y,s) triples.

If the direction of the edge-element is unknown, then the locus of points in parameter space representing circles which could have created this edge-element forms a right circular cone. In the presence of direction information, this locus is reduced to a line [Ballard, 1979]. As with line detection, circles which actually appear in the image will receive many votes; those which do not will receive few votes.

Arbitrary Shapes

The Hough technique can be extended to analytic shapes for which the mapping from edge-element to a locus of points in parameter space can be derived. Given certain assumptions about the meaning of "shape", we can also extend the technique to arbitrary, non-analytic shapes.

Consider a particular figure (e.g., an ellipse centered at (1,2) with its major axis parallel to the x-axis and of length 10, and its minor axis of length 5). Now, consider the set of figures which can be produced by translating, rotating, and uniformly scaling the original figure. For our purposes, all of these figures have the same shape.

The parameter space which captures this notion of shape is:

\[ P = \langle x,y,s,\theta \rangle \]

where \( \langle x,y \rangle \) is the origin of a local co-ordinate system, s is a scale factor, \( \theta \) is a rotation about \( \langle x,y \rangle \). This is the parameter space used in our generalized Hough Transform. Note that the Hough spaces developed above for lines and circles are sub-spaces of \( P \).

The key to all Hough techniques is the mapping from edge-element information to a locus of points in \( P \). We assume an edge-element operator which provides directional information. As seen above, this directional information can drastically reduce the image of the edge-element in \( P \). Our mapping, \( M \), depends strongly on the reliability of the edge-element direction.

Consider the hyperplane of \( P \) with

\[ \theta = 0, \ s = 1. \]

We represent the mapping from edge-element location and orientation to figure location directly in an

Figure 1: Geometry for Generalized Hough Transform

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \phi )</th>
<th>( R_{\phi_i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>{ r</td>
</tr>
<tr>
<td>1</td>
<td>\Delta \phi</td>
<td>{ r</td>
</tr>
<tr>
<td>2</td>
<td>2\Delta \phi</td>
<td>{ r</td>
</tr>
</tbody>
</table>

... ... ...

Figure 2: R-Table Format

"R-Table" (see Figures 1 & 2). The orientation of an edge-element is used as an index into this table, where are stored a set of \( \langle x,y \rangle \) vectors. When added to the \( \langle x,y \rangle \) location of the edge-element in the image, these vectors point to possible locations for the origin of a figure's local co-ordinate system (its reference point). This map is easy to build, given an original master shape.

The expansion of the R-Table mapping to cover the remainder of \( P \) is performed dynamically by our voting procedure, \( V \). This involves rotating the edge-element orientation before using it as an index into the R-Table, and scaling the R-Table entries thus formed before calculating the figure's hypothesized reference point.

3. Implementation and Experimental Results

The generalized Hough Transform described above has been implemented and tested on a variety of artificial images and has found application in the analysis of real aerial images. Experience to date indicates that the technique is robust, given that the edge-element operator used to generate local evidence for the shape can provide reliable information about edge-element direction.
R-Tables

The R-Table defines the mapping from edge-element information (position and orientation) into a hyperplane of parameter space. This mapping is derived from an explicit master shape, in the form of a sequence of boundary points. Typically, we sketch (or trace) a shape. In order to ease the pain of carefully drawing a particular shape, we customarily sample the master shape boundary rather coarsely and then fill in the intermediate points using a B-spline fit to these points [Riesenfeld, 1973]. An arbitrary reference point is chosen for the origin of the local co-ordinate system.

Now, for each point on the master shape boundary, we calculate the orientation of the boundary edge-element at that point and the vector from the boundary edge-element to the origin of the local co-ordinate system. This is exactly an R-Table entry. The current implementation of the R-Table consists of a list of entries, tagged with the edge-element orientation, containing a list of reference point vectors. Any scheme which associates edge-element orientation with reference point vectors will do.

Edge Detection

The examples shown below used a simple 3x3 Sobel edge-element finder. In general, this is satisfactory. When, as in one example below, this does not provide reliable edge-element orientation, performance deteriorates seriously.

Detection Criteria

For the purposes of these examples, the shape found by the generalized Hough Transform is determined by simply selecting the maximum value found in a smoothed (over a 3x3x3x3 window) Accumulator Array. This does the right thing when, as in most of our examples, the maxima in the Accumulator Array are sharp peaks. For more problematic, noisy situations, clustering in parameter space may be required.

Artificial Images

Figure 3a-3d illustrates a few of the features of the experimental implementation of the generalized Hough Transform. These artificial images provide controlled conditions for our testing. In Figure 3a and 3b we see that, as expected, the method has no difficulty in finding the central shape at arbitrary scale and orientation (The black dots show the shape, as drawn from the R-Table and the parameter choices which received the most votes in the Accumulator Array, the central black dot is the reference point.) In Figure 3c we see what appears to be the same shape, obscured by another. Figure 3d demonstrates that there is enough evidence for the desired shape to correctly determine its location, orientation, and scale.
All of these images, of course, have very clean edges and the 3x3 Sobel operator has no difficulty in correctly determining edge-element orientation. By way of contrast, see Figure 4a. In this image, which has been degraded by the addition of Gaussian noise of mean zero and standard deviation ten, the edge-elements found by the 3x3 Sobel operator are too short, and the noise hopelessly jumbles the orientation information. As a result, the generalized Hough Transform (which depends strongly on the accuracy of edge-element orientation) is unable to locate the shape. The guess shown is not much more than that - the Accumulator Array has no very strong peak, and we simply show the shape instance which received the most votes in a very close election.

To ameliorate the effects of noise, the image can be smoothed prior to applying the edge operator. As an experiment, the noisy image of Fig. 4a was smoothed by convolving it with a 5x5 template of ones. Next, the Hough algorithm was applied as before. Fig. 4b shows that, in this case, the edge estimates have been improved enough so that the shape is now correctly located.

Aerial Photographs

The location of arbitrary, non-analytic shapes is not merely of interest in artificial images such as that shown above. The original version of the shape found above came from the aerial image shown as Figure 5a. Even the experimental version of the generalized Hough Transform has no difficulty in locating the pond in this image, as shown in Fig. 5b.

4. Focus of Attention

One of the difficulties encountered in the application of this technique to real images, for the location of real
shapes, is that the area searched for evidence of boundaries (the application of the edge-element detector and the mapping from edge-element information to parameter space) and the size of the parameter space can quickly become very large. The solution to these problems is, of course, to attempt to focus attention where possible.

Where to Look

One way to focus attention during the application of this shape-finding technique is to constrain the area searched for evidence of the figure’s boundary [Russell and Brown, 1978; Russell, 1979]. In a system which routinely applies an edge operator over the entire image, this may not seem to be a solution (or even a problem). However, even after the edge-elements have been found, it is still necessary to apply the mapping to parameter space (one per desired shape). Our implementation includes the usual “bounding rectangle” limitation on the area in which edge-elements are to be found and mapped to parameter space. This improves performance significantly.

What to Look For

The second obvious way to focus attention is to constrain the objects being sought. Of course, a single application of the generalized Hough Transform concentrates on the location of a particular class of shape (that defined by the R-Table). In addition, it is usually possible to constrain the permissible values for some (if not all) of the parameters. Constraining the location of the reference point is related to the question of “Where to look”. Constraining the parameters of scale or rotation is also possible, and certainly worth doing. Sometimes, the unconstrained search for a particular shape (such as the pond in Figure 5) will result in almost complete information about the range of values to be considered in successive searches. For example, once the pond has been located (in parameter space, including location in the image, rotation and scale) map-like knowledge about this particular part of the world would allow the search for other shapes in the scene to be almost completely determined. Thus, our technique can both generate and benefit from such constraints.

Although our current implementation uses only a simple “bounding rectangle” constraint on the area of the image to be searched for boundary information, it is possible to combine information about the range of locations for the reference point, scale, and rotation. When all of these are sufficiently constrained, then the R-Table itself provides pointers to the locations to be searched in the image for edge-elements.

5. Conclusion

Shape is an important defining feature of many image objects, often the only useful feature. The key ideas behind the Hough Transform have been extended to produce a shape detection technique which performs well in the presence of occlusion, even for completely arbitrary, non-analytic shapes. As has been demonstrated, however, the technique depends strongly on the reliable estimation of edge-element orientation.

References


Ballard, D.H., Generalizing the Hough Transform to Detect Arbitrary Shapes, TR55, Computer Science Department, University of Rochester, October, 1979 (a); submitted to *Pattern Recognition*.


THE ROCHESTER NATURAL LANGUAGE UNDERSTANDING PROJECT

James F. Allen

1. Introduction

This report describes the natural language understanding project at the University of Rochester. It discusses the general philosophy behind our approach and describes our recent research as well as the directions we intend to follow in the next few years.

Our interests lie mainly in designing and constructing systems that can partake in extended dialogues with a user about some well defined topic or set of topics. Typical contexts in which such dialogues occur include:

- Information retrieval contexts, where the user must extract information that will be used to accomplish some goal (e.g., dialogues at an information booth in a train station or with a library information system).

- Advisor/assistance contexts, where the system instructs and/or aids the user in performing a complex task (e.g., assembling an air compressor or cooking a meal).

Our long range goal is to be able to function adequately in a naturally occurring dialogue, where one finds extensive use of indirect speech forms as well as partially-formed and ill-formed sentences. The system must have the ability to infer the user’s goals in the dialogue and, most importantly, to recognize and resolve small misunderstandings that arise frequently in any such dialogue.

Up to the present time, the group has mainly been concerned with contextual aspects of language analysis. This has included work on dialogue structure as it relates to the goals of the participants and the identification of the knowledge required in order to be able to partake in and understand dialogues on various topics. This coming year, we will be extending our efforts to include the parsing phases of analysis, which will be mainly directed by Steven Small, who will be joining us from the University of Maryland in the late spring.

2. Our View of the Problems

The analysis of natural language by computer has been studied extensively in recent years. However, much of the research has viewed natural language as a formal object and attempted to describe its structure in isolation from how language is used. Only quite recently has natural language been considered as a means of communication — a method of transferring information. As a means of communication, it is a highly efficient mechanism. It is so effective mainly because of the extensive use of context by the speakers and hearers. By context, we mean their shared knowledge of:

- the situation in which the communication occurs;

- the activities and facts that are being discussed; and

- the individual beliefs and goals of the participants.

Most computer models of language have consisted of fairly sophisticated syntactic and semantic analysis techniques that can extract a literal meaning from a fully specified, well formed sentence. These models have used contextual knowledge in a few ways, but we argue that context must be considered in a more general manner to obtain significant improvement in our understanding of how language is used for communicating and retrieving information. Contextual information has been used in the following senses by previous investigators:

- what can be inferred from general knowledge about the activities and facts that are being discussed.

- how the previous sentences in a dialogue establish the ability to: 1) choose the correct senses of the words in a new sentence, and 2) resolve references to previously mentioned concepts (e.g., using pronouns).

- what can be inferred from the setting of the actual utterance, and the shared knowledge and goals of the speaker and hearer, that allows sentences to be understood not in their literal sense.

Context in the first sense has been studied by summarizing everyday knowledge of common activities (e.g., [Schank and Abelson, 1975]). These summaries can be represented in relatively large scale descriptions of activities as a sequence of subactivities (scripts) or in smaller descriptions of an activity as a unit that changes the world in some way (an action). A sentence is then interpreted either by:

- identifying the situation it describes with a subactivity in a script and then inferring that other subactivities in the same script occurred (e.g., [Cullingford, 1977]): or

- identifying the situation it describes with an action, and then constructing a sequence of actions (a plan) that is inferred to be occurring (e.g., [Wilensky, 1978]).
Context in the second sense has been incorporated in most large systems to some extent. A typical treatment of pronominal reference (as well as other types of reference) is to break the analysis into two subproblems:

- analyze each sentence as it occurs for parts that could be referred to later (e.g. see [Webber, 1978]); and

- when a reference occurs, select its referent from the list of candidates using "real world knowledge" (e.g. see [Rieger, 1975; Hobbs, 1978]).

Techniques of word sense selection involve using "semantic" similarity measures between the words and concepts already considered and the senses of the ambiguous word (see [Rieger, 1975; Riesbeck, 1975; and Hayes, 1977]).

The final sense of context, the one we are most interested in, is quite different from the above two. In particular, it alone addresses the problems of analyzing fragmented and/or ill-formed input. The main problem it concerns, however, is that people seldom speak literally (and, of course, seldom have problems understanding others not speaking literally!). The type of knowledge needed here is knowledge of the communication setting in which the sentence occurred. We claim that the most important aspects of the setting are:

- what the speaker and hearer believe about each other's beliefs and goals; and

- what linguistic options are available at the given time.

Since these are relatively new issues in natural language processing (though certainly not new in philosophy—see [Grice, 1957] and [Austin, 1962]), perhaps some examples would be useful. The following dialogues could occur between a user of a public library and a natural language system that provides library information. While these examples are constructed, we shall see very similar ones in Section II that were actually collected in a similar situation.

A good answer to a question often provides more information than strictly required. It does not, however, provide too much information or provide information that is of no use to the person who made the query. Thus the person giving the answer must know what kind of information the questioner is after, and what he already knows about it. In this setting, the following might occur:

(1.1) User: What is the call number of War and Peace?

(1.2) System: IX321, but it is out at the moment.

Although the status of the book in the library was not explicitly requested, the system should provide it in order to be helpful. We claim that there are (at least) two essential facts that the system must know (or infer) to give this response:

- the user will probably want to take the book out; and

- the user is probably not aware that the book is out at present.

If either of these assertions were not true, the response above would seem inappropriate.

Similar problems can arise from the fact that people often communicate using sentence fragments. For instance,

(2.1) User: War and Peace?

(2.2) System: It is out at present.

Neither the syntactic form of the query nor the meaning of its words indicate what the response should be. The system needs to be able to infer enough about the user's beliefs and goals so that the mere mention of an object (e.g. a book title) is sufficient to identify what information is being sought.

Finally, other sentences should not be treated at face value. For instance:

(3.1) User: Do you know any books about making harpsichords?

Syntactically, this is a yes/no question about the system's knowledge; however, an answer of "yes" in the given situation would be quite inappropriate. Not all sentences beginning with "Do you know..." are intended in this indirect way however: the syntactic form can be used literally. To distinguish between literal and non-literal uses of sentences the system must have an idea of what questions are reasonable for the user to ask. This, in effect, requires knowing what goals the user is likely to be pursuing.

The ability to use knowledge about the user's goals and beliefs seems to offer the most promising advances toward more robust and general natural language systems. In particular, it alone addresses the issues of communication using fragmented and/or ill-formed sentences, as well as the recognition of the true intention behind indirect requests.
3. Previous Research

In our initial work on this problem, described in [Allen, 1979], we developed a system that recognized the intent of a speaker in a simple situation, and then used this knowledge to generate helpful responses. The basic components of the model included:

- a knowledge representation capable of maintaining and distinguishing between the goals and beliefs of multiple agents [Cohen, 1978];

- an analysis of utterances as actions that modify the world by changing the participants beliefs and goals [Perrault and Allen, forthcoming];

- a recognition component to identify the linguistic action performed by an utterance as well as the plan of the speaker that includes that action [Allen and Perrault, 1979]; and

- an analysis technique for using the plan recognized above with the knowledge of the speaker's beliefs that produce a set of topics that would constitute an appropriate (and helpful) response.

Let us briefly consider each in turn.

To represent the beliefs of different agents (users and the system), it is necessary to index each fact with information as to:

- who believes it is true;
- who believes it is false; and
- who does not know whether it is true or false.

This final case is important, for it indicates that the agent does not know whether the fact is true or not; thus he may engage in a dialogue to find out. The method we used for representing belief was to divide the knowledge base into partitions (see [Hendrix, 1975]), with each partition containing one agent's set of beliefs. Of course, things become more complicated than this since a major part of one agent's beliefs may be what he believes another agent believes. So partitions may be enclosed inside other partitions to indicate such nesting of beliefs.

Language can be viewed as goal-oriented behavior if we consider that utterances are produced by linguistic actions (speech acts) that are intended to effect the beliefs of the hearer. As a simple example, we can define an action as informing a person X that some proposition P is true by the following conditions:

- a prerequisite that the speaker believes P is true (in order to be sincere)
- an intended effect that the hearer believes P is true.

Thus the primary purpose of an inform is that the hearer acquire some new knowledge.

The recognition component has two tasks. It must identify the linguistic action (or actions) that are performed in saying the utterance, and it must identify the goals and plan that the speaker had that caused him to execute the linguistic action. Let us consider each in turn.

As argued by Searle [1975], we assume that all sentences (that are not idioms) must be taken literally at some level. It is only via the literal meaning that an indirect meaning can be inferred. Most literal linguistic acts can be identified from the syntactic mood of the sentence. For example, an imperative mood sentence indicates a request to do something.

The next task is to infer what plan the speaker is performing. We know that this plan should involve the observed linguistic action and expect that it will be addressing one of our expected goals of the speaker. Possible candidates for the speaker's plan can be synthesized using techniques strongly related to those used in current problem solving systems such as [Sacerdoti, 1975]. For instance, one simple plan construction strategy is backwards chaining, where a planner is given a goal and attempts to find an action to accomplish that goal. A simple plan inference technique would be to start with an action and try to infer what goal the agent is trying to accomplish. The plan inference techniques are not limited to such a simple method, however. In particular, there can be inferences up an abstraction hierarchy (such as in Sacerdoti), as well as inferences into knowledge structures such as scripts and plans [Schank and Abelson, 1975]. The plans inferred by these methods are continually evaluated as to how likely these are to actually be the speaker's plan by a set of plan inference heuristics (see [Allen and Perrault, 1979]).

The final component of our solution involves selecting an appropriate response to an utterance. Put in our terms, this means we must select the goals that we are going to accomplish by our utterance. Since we want to produce a helpful response, the goals we select should contribute to the successful execution of the user's plan. To ensure this, we examine the user's plan derived by the plan recognition and select from it:

- goals that the user will have trouble accomplishing by himself; and
- assumptions upon which this plan depends which we believe are not valid.
Our claim is that a response based on the above goals is both helpful and in many cases expected by the user. Let us consider examples similar to those given in the introduction, and see how the system derives the appropriate response. These examples are taken from transcripts collected at an information booth in a train station [Horrigan, 1977]. The clerk can be viewed as the linguistic interface between the user (a patron at the station) and a database of train arrival and departure information. Although this is a reasonably simple domain, the linguistic behavior observed is quite complex. Hence, it is an ideal domain for testing ideas about natural language interfaces to information retrieval systems.

**Example 1:** A Helpful Response

Patron: When does the Montreal train leave?

Clerk: 3:45 at gate 7.

It is fairly simple to see how the model explains the providing of more information than explicitly requested. The question above indicates that the patron executed the action of requesting the clerk to inform him of the departure time of the Montreal train. In this domain, the clerk also has expectations that patrons usually have goals of meeting or leaving on trains. The query about a departure time indicates that the patron's goal is to board the train to Montreal. (This is accomplished by the plan recognition component.) Once this plan is derived, the clerk can find that two key subgoals in boarding a train are knowing the departure time and the location. Since the clerk believes the patron knows neither of the above, his response addresses both goals.

**Example 2:** A Sentence Fragment

Patron: The 3:15 train to Windsor?

Clerk: Gate 10.

In this example, the patron does not utter a complete sentence; his utterance simply describes a train. However, given the clerk's expectations, we see the clerk can again easily infer that the patron's plan must be to board the 3:15 train to Windsor, for it is the only expected goal that could involve the train described. So once again, the clerk can identify two key subgoals of knowing the departure time and location. However, the initial question indicates that the patron already knows the departure time, thus only the departure location is given in the response.

**Example 3:** An Indirect Speech Act

Patron: Do you know when the Windsor train leaves?

Clerk: 3:15 at gate 10.

In this case the speech act performed appears to be a yes/no question concerning the clerk's knowledge. However, in the given setting, the clerk assumes that patrons already know that information clerks at train stations know departure times, so the literal goal indicated by the patron's query already holds. Since people do not speak for no reason, the clerk examines alternate possible goals. In particular, the clerk notices that if the literal goal is true, then it enables him to tell the patron when the train leaves. This action fits well with the expectation that the patron wants to board a departing train. Once this is recognized as the patron's plan, the response can be generated in the same manner as the above examples.

The system described above was implemented [Allen, 1979] and extensively tested in the train domain. This work was encouraging for the mechanisms of this system itself are entirely domain independent. In addition, only simple plan inference techniques, the equivalent of backwards chaining and abstraction hierarchies, were used in the solution. Considerably enhanced system performance could be expected by using domain specific knowledge. Thus, by changing the knowledge base of facts and the expected goals, domains of similar complexity could be easily handled. Our interests, however, lie in investigating more complex domains, and some interesting new problems arise. These will be described in the next sections.

4. Current Research Areas

We are currently pursuing both theoretical and practical issues. On the theoretical side, we are currently developing more adequate representations for actions, including complex linguistic actions. This work by necessity involves specifying a formalism in which temporal information can be expressed and reasoned about. We are also considering new formulations of belief and knowledge that allow for the representation of incomplete knowledge as well as for knowledge about other agents' knowledge.

More on the practical side, we are attempting to develop new heuristics to guide the plan recognition process. These heuristics will attempt to take advantage of the fact that the plans being recognized were actually constructed (by the speaker) with the intention of being recognized.

The final set of issues concern the actual system implementation. These concern how the overall system is to be organized, what role the parser plays with respect to the rest of the system, and in what form knowledge should be represented.

The remaining sections of this report will deal with these issues in more detail.
Representation of Actions

The class of actions that are described in conversations and also required to model linguistic actions is more general than found in current problem solving formalisms. For example, the action that counts as an intended request can be performed by a vast range of different literal linguistic actions. It is not feasible to simply catalogue the different methods as they are highly dependent on the current context and new forms are continually constructed and recognized. We saw some examples of indirect requests above.

Such problems are not particular to speech acts. For instance, one action such as hiding from someone can be performed by virtually any activity. For example, consider:

- crouching behind a car;
- moving to a new location; or
- standing still (in the case where one intended to move to another location where one would be seen).

All of these can be considered as the action of hiding as long as the agent has the intention of not being seen.

A formal model of action that can account for such behavior is currently being investigated by the author. Very briefly, it involves specifying a representation for temporal information and then defining actions as conditions on the world model over time. Any activity that then matches this description over time can be counted as the performance of the action. Thus the action of hiding occurs whenever the agent does some activity with the correct intentions. Similarly, a request occurs whenever the agent has the intention to get the hearer to do some other action, and furthermore, intends that the hearer recognize this first intention. This analysis of the linguistic actions was mostly developed in [Allen, 1979], but could not be readily expressed in the action representation used (which was based on STRIPS [Fikes and Nilsson 1971]). The new representation is well-suited to this type of analysis.

A major subissue of this endeavor is specifying a representation in which temporal information can be easily expressed. Such a representation, a formulation based on time intervals, is described in a report by the author [Allen, forthcoming] which also describes the action representation in detail.

Belief and Knowledge

The belief representation used in the previous system (see [Allen 1979], Cohen 1978) was adequate for simple domains, but had an inadequate treatment of beliefs about other agents' knowledge. Other formalisms recently suggested (e.g. [Moore 1979]) also suffer from the same weakness. One of the problems arises in representing a fact of the form

"John knows where Sam lives."

The representation of this fact must not insist that Sam's abode is known to the speaker. To clarify this point, if we know that Sam lives on 4th Street, then we could express the above assertion with a formula of the form

\[ \text{KNOW(John, ABODE(Sam)) = 4th Street} \]

(Assuming KNOW as a modal operator).

But if we don't know Sam's address, we can still believe the above assertion, even though we have no constant to place in the formula (where 4th Street was used). In all the formalisms mentioned above, this situation was represented by quantifying over the KNOW operator, i.e.

\[ (n) \text{KNOW(John, ABODE(Sam)) = n}. \]

This scheme works as long as we assume that there is a standard name for Sam's abode which will uniquely identify it. Unfortunately, this is not the case. The truth of the statement that John knows where Sam lives depends on what John needs to do with the information! For instance, if John is a census-taker, then 'Rochester' may be a sufficient description. However, if John is a friend coming to a party at Sam's, then John must know an address.

In other words, John knows where Sam is only if John has a description of where Sam lives that provides him with enough information in order to do whatever he needs to do with Sam's address. To express this, we need a formalism that can reason about the form of its descriptions.

We have been studying logical formalisms which can reason about their own syntax. In other words, the logic has a quotation mechanism which is quite powerful, and a predicate 'Tr' which relates true sentences to the same sentence quoted. This can be done in an intuitively satisfactory, consistent manner, which is described in [Perry, 1980].

With the ability to reason about the form of other agents' beliefs, we can start to consider some of the knowledge issues mentioned above. Current issues under investigation involve formulating plans to acquire knowledge, and simulating the reasoning of other agents given knowledge about their beliefs. Some of this work is reported in [Haas, 1980].
New Heuristics for Plan Recognition During Conversations

Recognizing the other person’s plans and intentions during a conversation is a task quite different from what has been previously studied as plan recognition. In one way, it is a far more difficult and complex task, yet when viewed from another light, it may actually be a simpler task. It is a more complex task than the plan recognition work done by [Schmidt et al., 1979] for they may observe many actions in sequence before committing themselves to a plan. In a conversation, of course, one observes only one action, the other’s utterance, from which one must recognize enough of the other’s plan to produce an intelligent response.

On the other hand, it should be a simpler task because:

- the speaker is a “rational” person and so would only say something that described his intentions sufficiently to be understood; and

- there is a coherence between sentences in a dialogue, and thus there may be very strong expectations developed about the user’s plan as the dialogue continues.

Much work remains to be done exploiting these aspects before we can overcome the disadvantage of being allowed to observe only a single action at any one time. Let us consider these points in turn.

In our current investigation, the assumption that the speaker constructs his utterances rationally is exploited by the following heuristic controlling the search strategy. If we reach a choice point between two possible plans for the speaker, where

- one plan enables a large number of mutually exclusive inferences that could be made about the speaker’s goals, and

- the other introduces no uncertainty as to what inferences to make next,

then favor the latter (i.e. the one that introduces the least uncertainty). This can be extremely useful in reducing the search space of possible plans. It is justified in the following way: since the speaker is rational, he could not expect the hearer to understand him if it required an extensive search through possible interpretations. Thus he will design his utterance so that there is a relatively direct inference path to his intentions.

The success of this heuristic, of course, depends on the speaker and hearer having accurate models of each other. It is reassuring to note that this fact is also essential to enable coherent conversation between people. In fact, in cases where a person does not have a good model of his audience, he tends to speak in a more precise and formal manner.

While this heuristic is still somewhat speculative, we hope to demonstrate its effectiveness by analyzing its performance over many dialogues. If it proves to be useful, it will be a major contribution to reducing the search space of a very complex task.

Another aspect we are investigating, namely the exploitation of coherence over an extended dialogue, may prove to be more essential in enabling plan recognition. When a dialogue is initiated, there is considerable uncertainty as to what the goals of each participant are. However, the opening exchanges usually establish these goals reasonably explicitly. From then on, there are very strong expectations about the other’s goals, and hence strong expectations about what types of actions they will perform. This is captured by maintaining the user’s plan derived from the previous utterances, and using the outstanding subgoals in that plan as the expected goals of the new utterance. While this has been an effective technique in the limited scale domains we have considered so far, techniques have yet to be developed that are feasible in larger domains where the plans may become extremely complex.

The Proposed System

The natural language system currently under implementation is a collection of independent modules that communicate using message passing. The modules tend to be quite large themselves, and relatively few in number. The initial system involves three modules interacting with a common knowledge base. These are

- the parser, responsible for meaning analysis of input based on semantic and syntactic knowledge;

- the context manager, responsible for maintaining the discourse structure, identifying the referents of description, and for maintaining other contextual information;

- the plan recognizer, responsible for inferring goals of the speaker that are not explicit in the utterances. This is based on the work described above and in [Allen and Perrault 1979].

These modules will interact during the analysis of utterances, and cannot be viewed as operating in a simple sequence. This can be best demonstrated by an example -- the analysis of noun-noun modification in English. When a noun is used to modify another noun, there can be virtually no information provided as to how they are related. For example, depending on the context, the noun phrase “the Windsor train” may indicate the
train with destination Windsor,' 'the train coming from Windsor,' 'the train owned by the city of Windsor,' or 'the train in Windsor at the particular moment.' This list can be extended almost indefinitely by constructing new contexts.

One of the few analysis techniques used to solve this problem is to find a relationship between Windsor and a train that is relevant to the context (the user's plan). But this relationship may not be discovered until arbitrary amounts of deduction have been done by the rest of the system. Thus, the parser cannot finish its analysis until the rest of the system has performed deductions that must be initiated by output from the parser!

To overcome these difficulties we must view the parser as producing incremental outputs as the analysis proceeds. The system does not wait for a 'complete meaning' of the utterance to be produced. In fact, in many cases, for instance when analyzing sentence fragments, a 'complete meaning' may never be produced. Thus the parser is but one part of a complex system that attempts to understand the user's intentions. In particular, it is solely responsible for what inferences can be made about the user's intentions based on the syntactic form of the utterance and the word meanings used. It will not communicate any messages that resemble parse trees, as no other module could interpret them. It will produce assertions about what the speaker is trying to do (i.e. its literal speech act), what objects are referenced, what facts are conveyed and what facts are assumed (i.e. presuppositions), as well as what it can analyze about the dialogue structure.

The major advantage of this design is that the system does not wait for the parser to complete before it can start performing inferences such as identifying referents of noun phrases and recognizing the plan of the speaker. Thus some response would be generated whether or not the parser could make sense of the sentence. Alternatively, the inferences made by the rest of the system may provide the parser with strong enough expectations that an initially undecodable sequence of words becomes understandable.

The factual knowledge that is used by these modules is stored in a central knowledge base. The kernel of the knowledge base provides a context mechanism that allows the system to operate in different modalities. For instance, operations can be specified relative to a particular agent's beliefs at a certain time as well as with respect to a set of assumptions made about the world.

The kernel also allows the designer of the knowledge base to specify what inferences can be made during the pattern matching/retrieval process. At present, we are using this to define some of the deductions commonly found in semantic network formalisms. This includes the inheritance of properties on a ISA hierarchy (set membership and set inclusion), default values and functional roles as in case frames. We are also defining the temporal reasoning mechanism (as described above), and a facility to reason about the equality of descriptions relative to contexts. When we have completed this, all these features will appear to be "built-in" to the pattern matching mechanism of the knowledge representation. All other deductions that are required must be made by the inference modules themselves.

Except for the features described above, the kernel makes no commitment as to the style of the representation. Depending on the application, an appropriate representation scheme (e.g. conceptual dependency, partitioned semantic nets, predicate logic) could be defined, hopefully taking advantage of the features offered in the kernel.

At the time of this report (Fall 1980), implementation is just beginning. The message passing facility between LISP programs (running on a VAX 11/780) and the kernel of the knowledge base are under construction. These tools will be exercised and elaborated upon by implementing a series of toy systems. The initial system will be designed to operate in multiple domains. At present, we intend to consider more dialogues from the train station domain, as well as a set of dialogues between a computer operator and users concerning the mounting of tapes, restoring files and other computer operations.

5. Concluding Remarks

The natural language understanding project at Rochester is attempting to design and construct systems that can partake in dialogues that have much of the complexity found in naturally occurring dialogues. In the investigation, we are searching for analyses that are justifiable on theoretical grounds independent of a particular implementation, but yet can be adapted and used in actual systems.

The group currently consists of James Allen, Gary Cottrell, Michel Denber, Alan Frisch, Andy Haas, Mark Kahrs and Donald Perlis. In late spring, 1981, Steven Small will join the group to strengthen the parsing efforts. Our group is fortunate also to have a large number of other interested parties in the University, both within the Computer Science Department as well as in other departments, particularly Philosophy and Psychology. Of particular note are the other artificial intelligence faculty in Computer Science, which includes Dana Ballard, Chris Brown, Jerome Feldman, Ken Sloan, and, starting in the spring of 1981, Pat Hayes.
References


Wilensky, R., "Understanding Goal Based Stories," RR140, Department of Computer Science, Yale University, 1978.
AN INFORMATION-THEORETIC APPROACH TO TIME BOUNDS FOR ON-LINE COMPUTATION
Wolfgang J. Paul, Joel I. Seiferas, and Janos Simon

Introduction
Static, descriptive complexity (program size) [18, 9] can be used to obtain lower bounds on dynamic, computational complexity (such as running time). We describe and discuss this "information-theoretic approach" in the following section. Paul introduced it in [14], to obtain restricted lower bounds on the time complexity of sorting. We use the approach here to obtain lower time bounds for on-line simulation of one abstract storage unit by another. A major goal of our work is to promote the approach.

Our main results show that more points of access into storage can save significant time. The storage units we consider are multihead variants of the ordinary one-dimensional "Turing machine tape", and the points of access are the tape heads. Our bounds complement earlier results by Rabin [17], Aanderaa [1], Hennie and Stearns [8], Hennie [7], and Pippenger and Fischer [16] and recent results by Loui [12] and Paul and Reischuk [15].

Aanderaa's result considers on-line simulation of a single-head one-dimensional tapes (or even pushdown stores) by $h'$ such tapes, for $h' < h$ (inadequate number of tapes). Although his argument does not yield any superlinear lower bound for the worst-case time $T(n)$ to handle $n$ commands, it does show that such a simulation is impossible in real time. This generalizes Rabin's earlier result for $h = 2, h' = 1$. Our main results, Theorems 1 and 2 below, include a multidimensional version of Aanderaa's result and a version for "tree" tapes. The proofs are entirely new, however, and do yield superlinear lower bounds.

In our corollary to Theorem 3 below, we show how to strengthen Aanderaa's result about tapes to a result about heads. In fact both Aanderaa's result and our higher-dimensional and tree versions hold even if the $h'$ simulating heads can be on the same tape and are allowed to jump to each others' positions. Even if the $h'$ simulating heads can be on a $d'$-dimensional tape for $d < d'$, Theorem 1 still gives a superlinear lower bound for the time to simulate an $h$-head $d$-dimensional tape unit. All this supports the general conjecture that nothing else can make up for an inadequate number of points of access into storage.

Now suppose that every $d$-dimensional tape can be simulated on-line in time $T(n)$ by a similar tape with just $h'$ heads ($h'$ fixed). By the corollary to Theorem 1 below, $T(n)$ has to be $\Omega(n^{1+1/d-\epsilon})$ for a relatively small ($\epsilon$ just barely greater than $2/(d^2 + d)$). In contrast, Hennie and Stearns' analogous upper bound for $d = 1$ is only $T(n) = O(n \log n)$. In Theorem 4, we show that every $d$-dimensional tape unit can be simulated on-line by a similar unit with just two heads in time $T(n) = O(n^{1+1/d-\beta})$, where $\beta$ is very small ($\beta = 1/(d^2(d-1)+d)$). (In fact, one of the two heads need only access a separate one-dimensional tape.) This shows that the lower bound in Theorem 1 is a rather good one.

Suppose, analogously, that every tree tape can be simulated on-line in time $T(n)$ by a similar tape with just $h'$ heads ($h'$ fixed). By Theorem 2 below, $T(n)$ has to be $\Omega(n \log n / \log \log n)$. On the other hand, Paul and Reischuk's corresponding upper bound is $O(n \log n)$; so the lower bound is nearly optimal. Moreover, the lower bound applies even if only tapes with $h'+1$ heads are to be simulated.

Hennie's result considers on-line simulation in time $T(n)$ of an $h$-head $d$-dimensional tape unit by an $h'$-head $d'$-dimensional tape unit, for $d' < d$ (inadequate dimension). His argument yields a superlinear lower bound proportional to $n^{1-1/d+1/d'}$ [8], regardless of $h$ and $h'$. For $d' = 1$, Pippenger and Fischer have a matching upper bound, even for $h'$ fixed. By the corollary to Theorem 1 below, there is no such matching upper bound for $d' > 1$ and $h'$ fixed; in this sense, the Pippenger-Fischer result cannot be generalized. For $h'$ not fixed, however, Loui has a corresponding upper bound $O(n^{1-1/d+1/d' + \epsilon})$ for any $\epsilon > 0$.

The Information-Theoretic Approach
Lower bounds on inherent worst-case computational complexity are notoriously easy to conjecture but hard to prove. One reason is the difficulty in finding sufficiently hard inputs for each different algorithm. The worst case for one algorithm might be expedited as a "special case" by another. If we can find inputs not susceptible to handling as special cases, then we might be able to convert our intuitions to proofs more easily.

One way to handle an input efficiently as a special case is to find and work with a much smaller description of the same input. We can prevent this sort of special handling if we provide inputs which are suitably incompressible. The work of Kolmogorov [9] and Solomonoff [18] shows how to make this precise in a robust way and that suitably incompressible streams of input data are abundant.
We have discovered that such an information-theoretic approach is particularly useful for proving lower bounds on the complexity of simulating abstract storage units on-line. The incompressibility forces simulators to use a lot of space and hence to spend a lot of time retrieving distant information. The approach is responsible for our main new results, Theorems 1 and 2 below.

The information-theoretic approach also serves as a rigorous, yet natural, tool equivalent to vague intuitions already in limited use. (In this sense, its value is analogous to that of nonstandard analysis.) These potentially valuable intuitions have not been cultivated much in the past, because conversion to rigorous proofs seemed so difficult. Rare successful conversions of this sort were performed by Hennie [7] and Aanderaa [1]. At certain points in their proofs, it seems that the argument should be able to proceed for any "typical" or "random" input sequence; but their proofs capture this intuition only with great effort, by counting aggregates of input sequences, essentially to show that not all can fail to be sufficiently "typical". Following Kolmogorov, the new approach in such situations is to look at a particular sequence which is random in the rigorous, domain-independent sense that it is incompressible. The effect is to remove obscuring domain-dependent counting from such proofs, and instead simply to cite the result of the one simple counting argument which shows that there are incompressible strings. The resulting simplification of Aanderaa's proof is presented as the proof of Theorem 3 below.

Descriptive Complexity [9]

We wish to define the descriptive complexity of a tuple of binary strings given another tuple of binary strings. We will use the symbol $\#$ to separate the components of our tuples. Any computable partial function $F : \{0, 1, \#\}^* \rightarrow \{0, 1\}$ can be viewed as a relative description scheme, in terms of which we can define a relative descriptional complexity $K_F : \{0, 1, \#\}^* \rightarrow \{0, 1\}$ as $K_F(x | y) = \min\{|d| ; d \in \{0, 1\}^* \text{ and } F(d\#y) = x\}$.

Because there is a "universal" computable partial function, there is some $F_0$ for which

$$\forall x \exists c \forall z : y \leq K_{F_0}(x | y) \leq K_F(x | y) + c_F.$$  

Except for an additive constant, therefore, $F_0$ is as succinct a relative description scheme as any; so we define the relative descriptional complexity $K : \{0, 1, \#\}^* \rightarrow \{0, 1\}$ by $K_F(x | y) = K_{F_0}(x | y)$. We define $K(x)$ to be simply $K(x | \lambda)$, where $\lambda$ is the null string.

Since there are $2^n - 1$ possible shorter descriptions $d$, we can be sure that $K(x) \geq |x|$ for some binary string $x$ of each length. Such strings are incompressible. For each $y$, similarly, $K(x | y) \geq |x|$ holds for some binary string $z$ of each length.

Abstract Storage Units

An abstract storage unit is an (infinite-state) "sequential machine" $S : \Sigma^* \rightarrow \Delta$ with finite input alphabet $\Sigma$ (commands), finite output alphabet $\Delta$ (responses), and internal state set $\Sigma'$ (command histories). A deterministic automaton (finite-state machine with access to some storage unit of its own) simulates an abstract storage unit (the "virtual" unit) if its input-output behavior matches that of the simulated storage unit. (Because each command's response must precede the next command, such simulations are said to be on-line. We consider only on-line simulations, so we choose to shorten "on-line simulation" to just "simulation".)

For a deterministic automaton to compute the output of a storage unit in time $T(u; n)$, it can simulate the storage unit in time $T(u; n)$. Real time means time $T(u; 1)$ is $O(1)$ (bounded by a constant), and time $T(n)$ means time $T(1; n)$.

We use the rule $S(u, w) = S(u, v)S(w, w)$ to extend storage unit $S$ to a function on $\Sigma^* \times \Sigma^*$. Two command histories $u, v \in \Sigma^*$ are equivalent for $S$ if $S(u, w) = S(v, w)$ for every $w \in \Sigma^*$.

Let us note that the tape units under consideration here do qualify as abstract storage units. For a pushdown store, each input command is either "push 0", "push 1", "test for emptiness", or "pop". The corresponding output responses are 0, 1, whether the store is empty (0 for yes, 1 for no), and what symbol (0 or 1) gets popped (0 if there is nothing to pop, respectively). A counter is a pushdown store without the command "push 1".

For an $h$-head $d$-dimensional tape unit, each input command is of the form, "Write the symbol $a'$ beneath tape head number $i$", and then shift that head up $j$ positions in dimension number $k$, where $1 \leq i < h$, $-1 \leq j \leq 1$, and $1 \leq k \leq d$. If we admit head-to-head jumps, then there are also commands of the form, "Reset head number $i$ to the position of head number $j$", where $1 \leq i < h$ and $1 \leq j \leq h$. In either case, the corresponding output response indicates what symbol tape head number $i$ is left scanning after the command is "executed". We assume that initially such a tape unit has all heads coincident and the symbol 0 ("blank") written at every tape location and that there is at least one other symbol, 1, in the tape alphabet.

The commands to a (binary) tree tape unit resemble
the ones to a multidimensional tape. The difference is that the possible shift "directions" are "parent", "left
child", and "right child". Just as there is no need to have a left end to a one-dimensional tape, there is no need to have an ultimate root to a tree tape; so that it provides no special information, however, the initial head location is required to be a left child all of whose ancestors are left children of their parents.

Note that several abstract storage units can be combined into one. The composite command alphabet is a disjoint union of the individual command alphabets.

An abstract storage unit with sufficiently atomic commands needs only a binary response alphabet ∆ = {0, 1}. Note that simulation of such a storage unit amounts to what is usually called an on-line language recognition problem, with 1 signalling "acceptance so far" and 0 signalling "rejection so far".

Inadequate Access to Multidimensional Tapes

Our first lemma demonstrates the effect of inadequate redundancy in a relatively limited-access representation of multiple-access data. For this lemma, it is enough to consider string data.

Lemma 1. Let X be a set of k strings z(1), ..., z(k), each of length m; and let z = z(1) ... z(k). Let Y be a set of k' strings, each of length m'. If z is incompressible, then there is an h-tuple (z(1), ..., z(h)) of strings from X such that K(z(1) # ... # z(h) | y(1) # ... # y(h')) > m/4 for every h'-tuple (y(1), ..., y(h')) of strings from Y, provided m', m/ log k, and (mk)/(m'k'h') are large enough in terms of h and h'.

Corollary. In addition, let the superset Z of Y be the set of all strings z for which K (z | y) is a small enough fraction of m for some y in Y. For the same h-tuple (z(1), ..., z(h)) as above, we still get K(z(1) # ... # z(h) | z(1) # ... # z(h)) > m/5 for every h'-tuple (z(1), ..., z(h)) of strings from Z.

Proof of Lemma 1: Suppose, to the contrary, that for every such h-tuple, there is such an h'-tuple such that K(z(1) # ... # z(h) | y(1) # ... # y(h')) ≤ m/4. To reach a contradiction, we show that K(z) < |z| if the parameters satisfy their constraints.

The number of h-tuples from X is k^h, while the number of h'-tuples from Y is only k'h'. hence, there must be some one such h'-tuple (y(1), ..., y(h')) which works for at least p = k'h'/k^h distinct such h-tuples. The number of distinct components of these h-tuples must be at least q = [p^h/k] ≥ k/k^h; let z(i_1), ..., z(i_q) be q such components.

To describe z, we can describe z(i_1), ..., z(i_q) in terms of y(1), ..., y(h') and provide the rest of z literally. For each j (1 ≤ j ≤ q), z(i_j) appears, say as z(i_j) in some h-tuple (z(1), ..., z(h)) for which K(z(1) # ... # z(h) | y(1) # ... # y(h')) ≤ m/4, say via shortest description δ_j. Let ẑ be obtained from z by omitting z(i_1), ..., z(i_q). We can describe ẑ by providing the following string, prefaced by O(1) bits of explanation (essentially an appropriate formalization of the current discussion):

\[ y(1) \circ \ldots \circ y(h') \circ i_1 \circ d_1 \circ i_2 \circ \ldots \circ i_q \circ d_q \circ y(q) \circ \hat{z}, \]

where the mapping from w to ŵ is the string homomorphism on \{0, 1\}^* with δ̂ = 00 and T̂ = 11, and where o = 01 and * = 10 serve as distinct delimiters. (Each number is given in binary.) It follows that

\[
K(z) \leq O(1) + h'(2m' + 2)
\]
\[
+ q\left((2 \log k + 2) + (2m'/4 + 2) + (2 \log h + 2)\right)
\]
\[
+ |z| - q m
\]
\[
= |z| - q m/2 + q(6 + 2 \log k + 2 \log h)
\]
\[
+ 2h' m' + 2h' + O(1)
\]
\[
\leq |z| - q m/3 + 3 k' m' + O(1)
\]
(provided m' ≥ 2 and m/6 ≥ 6 + 2 log k + 2 log h)
\[
\leq |z| - k' m' + O(1)
\]
(provided m' is large enough).

Lemma 2. For h' < h, suppose an h'-head d-dimensional tape unit with head-to-head jumps can simulate an h-single-head-tape d-dimensional storage unit in time T(u, r). For each sufficiently large r, there is a command sequence u_0 of length r such that the following holds for u = u_0 and for every longer command sequence u equivalent to u_0:

Either \( T(|u|) \log T(|u|) = \sum \left( \frac{d l - h'/h + e r}{r (d + 1) (1 - h'/h)} \right) \),

or \( T(u; r) = \sum \left( \frac{(d l - h'/h + e r)}{r (d + 1) (1 - h'/h)} \right) \)
The implicit multiplicative constants here* depend only on $d, h, d', h'$, and the size of the simulator's tape alphabet (but not on the particular simulation).

Proof: A similar lemma with conclusion $T(u; r) = \Theta(r^{d/d'})$ can be used in Hennie's lower bound argument for inadequate dimension ($d < d$). In both cases, we use the initial sequence $u_0$ to write a "sufficiently incompressible ball" $B$ of radius $r/2$ and to send all the virtual heads to its center. (We use the "head shift metric": $B$ consists of those virtual tape positions within $|r/2|$ head shifts of its center. The "volume" $V((r/2))$ of such a ball is the number of these tape positions. Similarly, $V'(r)$ is the volume of a ball of radius $r'$ on a simulator tape.) In Hennie's case ($d' < d$), a simple volume argument suffices: $k! \cdot V'(T(u; r)) = \Theta(T(u; r)^d)$ (the accessible representation volume) has to be at least proportional to $V((r/2)) = \Theta(r^d)$ (the volume of $B$). In the case of adequate dimension ($d' \geq d$), the argument has to be more sophisticated, since small radius is of concern. On the other hand, small radius prevents much redundancy, provided $d'$ is not too much smaller than $d$. It is this lack of redundancy that will create headaches for a simulator with an inadequate number of heads ($h' < h$).

Lemma 1 above was designed to capture the effect of inadequate redundancy in a relatively limited-access representation of multiple-access data. Our proof will exploit that lemma.

To select our ball $B$ of radius $r/2$, we choose a parameter $s$ ($1 \leq s \leq r$) and pick out $k = \Theta(V'(r)/V(s))$ disjoint subballs, each of radius $s$. In each of these subballs, we store (in some canonical manner) a string of length $m = \Theta(V(s))$, chosen so that some concatenation of these $k$ strings is incompressible. Let $X$ be the set of these $k$ strings.

Let $u$ be $u_0$ or any longer command sequence equivalent to $u_0$. In the representation by the simulator at the end of the initial command sequence $u$, consider the balls of radius $T(u; r)$ centered at the simulator heads. Choose a parameter $t$ ($1 \leq t \leq T(u; r)$) and cover these balls with $\Theta(V'(T(u; r))/V'(t))$ balls of radius $\Theta(t)$ such that each subball of radius $t$ lies entirely within a member of the cover. While they last, select and combine pairs of cover members both having nonblank volume less than $V'(t)/\log T(u)$. This reduces the number of cover members to $k' = \Theta(T(|u|)/\log T(|u|)/V'(t))$. (We were led to this economy by a weaker suggestion from M. Loui, A. Meyer, and M. Sipser.) For each uncombined member, select a depth-first listing (including shifts) of its contents. For composite members, list only the nonblank contents, with explicit addresses. (Even for absolute addresses, $O(\log T(|u|))$ bits will be enough.) Let $Y$ be the set of these $k'$ strings, each unambiguously padded out to length $m' = \Theta(V'(t))$. As in the corollary to Lemma 1, let $Z$ be the set of strings $z$ for which, for some $y$ in $Y$, $K(z | y)$ is a small fraction of $m$. Note that this set includes a description of the contents (even including any head positions) of each member of the cover at any possible time within the next $T(u; r)$ steps, provided $T(u; r)$ is a sufficiently small fraction of $m$. (Include the sequence of at most $T(u; r)$ writes, shifts, and jumps performed, and the relative location of each simulator head within distance $T(u; r)$ of the member.)

We show now that $T(u; r) = \Omega(rt/s)$, provided $V(s)$ is a large enough multiple of $T(u; r)$ (which does imply that $T(u; r)$ is a small fraction of $m = \Theta(V(s))$) and $V'(t)$ is a small enough fraction of

$$\frac{V(r)}{(T(|u|)/\log T(|u|))^{1-\frac{1}{d'/d}}}.\]$$

(For there to be no such $s \leq r$, we would have to have $T(u; r) = \Omega(V(r))$, which would already imply the second conclusion, since $V(r) = \Omega(r^d)$ and $T(|u|) \geq |u| \geq r^d$. For there to be no such $t \geq 1$, we would have to have $T(|u|)/\log T(|u|) = \Omega(V(r)^{1/h'})$, which would already imply the first conclusion, since $V(r) = \Omega(r^d)$.) So it is safe to assume such $s$ and $t$ exist.) If, in addition, $t/s$ is large enough (no loss of generality, since the assertion $T(u; r) = \Omega(rt/s)$ is trivial otherwise), then it works out that $m', m'/\log k$, and $(mk)/(tm'^{d'/d})$ are large enough for Lemma 1 and its corollary. In this case, let $Z = (Z_1, .. , Z_h)$ be the guaranteed $h$-tuple of strings from $X$. Consider the following $r$ commands:

[r/2] commands: Send the virtual heads to the subballs where $z_1, .. , z_h$ are stored.

[r/2] commands: Repeatedly, in $O(s)$ commands, make an inquiry requiring more than $t$ simulator steps. If there were ever no such inquiry, then we could construct $x_1, .. , x_h$ from the simulator and its "radius-$t$ instantaneous description" at that time. For some $h$-tuple $(x_1, .. , x_h)$ of strings from $Z$, then, an upper bound for $K(x_1 | .. | x_h)$ would depend only on the simulator. But this would contradict the corollary's assertion that $K(x_1 | .. | x_h)$ exceeds $m/5$, provided $r$ is so large that $m = \Omega(V(s)) = \Omega(T(u; r)) = \Omega(r)$ is sufficiently large.

---

*"$O$" means "at most some constant times", "$\Omega$" means "at least some constant times", and "$\Theta$" means both.
It follows that $T(u; r) = \Omega((r/s)^{t})$.

We get our strongest conclusion above if we choose $s$ as small as permitted ($V(s) = \Theta(T(u; r))$, or $s = \Theta(T(u; r)^{1/d})$ ) and $t$ as large as permitted

$$(V'(t)) = \Theta \left( \frac{r^{d}}{(T(|w|) \log T(|w|))^{2/d}} \right)^{2/(1-d/h)}$$,

or $t = \Theta \left( \frac{r^{d}}{(T(|w|) \log T(|w|))^{2/(1-d/h)}} \right)$. Solving $T(u; r) = \Omega(r/s)$ for $T(u; r)$ gives the desired lower bound. •

Theorem 1. For $h' < h$, suppose an $h'$-head $d'$-dimensional tape unit with head-to-head jumps can simulate an $h$-single-head-tape $d$-dimensional storage unit in time $T(n)$. For each sufficiently large $n$, $T(n) = \Omega(n^{1+\alpha}/(log n)^{\beta})$, where

$$\alpha = \frac{(d/d' - 1/d)(1 - h'/h)}{(d+1)(1 - h'/h) + (d/d')(h'/h)^{d}/h)}$$

$$\beta = \frac{(d/d')}{(d+1)(1 - h'/h) + (d/d')(h'/h)^{d}}$$

In particular, $T(n) = \Omega(n^{1+\alpha})$ for some $e > 0$ if $d' < d^2$; if $d' = d \geq 2$ and $h' = h - 1$, then $T(n) = \Omega(n^{1+\alpha})$ for any $e < (1 - 1/d)(d + 1)$. The implicit multiplicative constants here depend only on $d$, $h$, $d'$, $h'$, and the size of the simulator’s tape alphabet.

Corollary. Suppose that every $d$-dimensional tape unit can be simulated in time $T(n)$ by an $h'$-head $d'$-dimensional tape unit with head-to-head jumps ($h'$ fixed). Then $T(n) = \Omega(n^{1+\alpha}/(log n)^{\beta})$ for every $\gamma > (1/d' + 1/d)/(d + 1)$. In particular, $\gamma$ can be smaller than $1/d'$ if $d' < 1$, and $T(n) = \Omega(n^{1+\alpha}/(log n)^{\beta})$ for every $\gamma > 2/(d^2 + d)$ if $d' = d$.

Proof of Theorem 1: For convenience, denote the long expression in Lemma 2 by $F(r, T(n) \log T(n))$.

Consider any large enough $n$. The conclusion holds if $T(n) \log T(n) = \Omega(n^{\alpha}/\beta)$, since $h/h' > 1 + \alpha$; so suppose $T(n) \log T(n) \neq \Omega(n^{\alpha}/\beta)$. Take $r = \Theta(n^{1/\alpha})$ with $r^{d'} < n/2$. Choose $u_0$ as in Lemma 2, and inductively cite that lemma while $|u_0 \ldots u_i| \leq n$ to obtain $u_{i+1}$ of length $\Theta(r)$ such that $u_0 \ldots u_i u_{i+1}$ is equivalent to $u_0 \ldots u_i$ and such that the simulator requires $\Omega(F(r, T(u_0 \ldots u_i) \log T(u_0 \ldots u_i))) = \Omega(F(r, T(n) \log T(n))) = \Omega(F(r, T(n) \log T(n)))$ steps to handle $u_{i+1}$ following the initial command sequence $u_0 \ldots u_i$.

The first conclusion in Lemma 2 cannot hold, because it would imply $T(n) \log T(n) \geq T(|u_0 \ldots u_i|) \log T(|u_0 \ldots u_i|) = \Omega(r^{d'/h'}) = \Omega(n^{h'/h'})$. Therefore, $T(n) \geq T(u_0; n)$. Thus, $\Omega(n^{1-1/d} F(r, T(n) \log n)) = \Omega(n^{1-1/d} F(r, T(n) \log n))$. Solving for $T(n)$ gives the theorem. •

Inadequate Access to Tree Tapes

First we adapt Lemma 2 above to take advantage of the acyclic nature of tree tapes.

Lemma 3. For $h' < h$, suppose an $h'$-head tree tape with head-to-head jumps can simulate an $h$-single-head-tree tapes in time $T(u; r)$. For each sufficiently large $r$, there is a command sequence $u_0$ of length $2^r$ such that the following holds for $u = u_0$ and for every longer command sequence $u$ equivalent to $u_0$:

Either $T(|u|) = \Omega(2^{h'/h'})$,

or $T(u; r) \log T(u; r) = \Omega\left((r - (h'/h) \log T(|u|))\right)$.

The multiplicative constants here depend only on $d$, $h$, $d'$, $h'$, and the size of the simulator’s tape alphabet.

Proof: To select our ball $B$ of radius $r/2$, we choose a parameter $s (1 \leq s \leq r)$ and pick out $k = \Theta(V(d)/V(s))$ disjoint complete subtrees, each of depth $s$. In particular, $\gamma$ can be smaller than $1/d'$ if $d' < 1$, and $T(n) = \Omega(n^{1+\alpha}/(log n)^{\beta})$ for every $\gamma > 2/(d^2 + d)$ if $d' = d$.

Proof of Theorem 1: For convenience, denote the long expression in Lemma 2 by $F(r, T(n) \log T(n))$.

Consider any large enough $n$. The conclusion holds if $T(n) \log T(n) = \Omega(n^{\alpha}/\beta)$, since $h/h' > 1 + \alpha$; so suppose $T(n) \log T(n) \neq \Omega(n^{\alpha}/\beta)$. Take $r = \Theta(n^{1/\alpha})$ with $r^{d'} < n/2$. Choose $u_0$ as in Lemma 2, and inductively cite that lemma while $|u_0 \ldots u_i| \leq n$ to obtain $u_{i+1}$ of length $\Theta(r)$ such that $u_0 \ldots u_i u_{i+1}$ is equivalent to $u_0 \ldots u_i$ and such that the simulator requires $\Omega(F(r, T(u_0 \ldots u_i) \log T(u_0 \ldots u_i))) = \Omega(F(r, T(n) \log T(n))) = \Omega(F(r, T(n) \log T(n)))$ steps to handle $u_{i+1}$ following the initial command sequence $u_0 \ldots u_i$. (The first conclusion in Lemma 2 cannot hold, because it would imply $T(n) \log T(n) \geq T(|u_0 \ldots u_i|) \log T(|u_0 \ldots u_i|) = \Omega(r^{d'/h'}) = \Omega(n^{h'/h'})$. Therefore, $T(n) \geq T(u_0; n)$. Thus, $\Omega(n^{1-1/d} F(r, T(n) \log n)) = \Omega(n^{1-1/d} F(r, T(n) \log n))$. Solving for $T(n)$ gives the theorem. •
shifts) of just its \( \text{nonblank} \) contents. Let \( Y \) be the set of these \( k \) strings, each unambiguously padded out to length \( m' = \Theta(V'(3t)) \).

Unless one of the desired conclusions is conceded to hold, Lemma 1 and its corollary apply as before to show that \( T(u;r) = \Omega(r/s) \), provided that now \( V(s) \) is a large enough multiple of \( T(u;r) \) and \( V'(3t) \) is a small enough fraction of \( T(u; r) \).

\[
\frac{V(s)}{T(u; r)} < \frac{1}{\log_2 n}.
\]

(Again, there can fail to be such \( s \) or \( t \) only if one of the desired conclusions already holds.) We get the desired lower bound by solving \( T(x;r) = \Omega(r/s) \) for \( T(x;r) \) where \( s \) is as small as permitted \( \left( s = \log_2 T(x;r) + \Theta(1) \right) \) and \( t \) is as large as permitted \( \left( t = \Theta(r - (h'/h) \log_2 T(x)) \right) \).

Theorem 2. For \( h' < h \), suppose an \( h' \)-head tree tape with head-to-head jumps can simulate \( h \) single-head tree tapes in time \( T(n) \). For each sufficiently large \( n \), \( T(n) = \Omega(n \log n / \log \log n) \). The implicit multiplicative constant here depends only on \( h, h' \), and the size of the simulator's tape alphabet.

**Proof:** Consider any large enough \( n \). The conclusion holds if \( T(n) = \Omega(n \log n) \), so suppose \( T(n) \neq \Omega(n \log n) \). Take \( r = \log_2 n - \Theta(1) \) with \( 2^r < n/2 \). Choose \( u_0 \) as in Lemma 3, and inductively cite that lemma while \( |u_0 \ldots u_l| \leq n \) to obtain \( u_{i+1} \) of length \( \Theta(r) \) such that \( u_0 \ldots u_i u_{i+1} \) is equivalent to \( u_0 \ldots u_i \) and such that the simulation requires \( T(u_0 \ldots u_i) \) steps to handle \( u_{i+1} \) following the initial command sequence \( u_0 \ldots u_i \) where

\[
T(u_0 \ldots u_i) = \frac{T(u_0 \ldots u_i; r) \log T(u_0 \ldots u_i;r)}{T(n)}
\]

from which it follows that \( T(u_0 \ldots u_i; r) = \Omega\left(r^2/\log r\right) \).

(The first conclusion in Lemma 3 cannot hold, because it would imply \( T(n) \geq T(u_0 \ldots u_i) = \Omega(2^{h'/h'}) = \Omega(n^{h'/h'}) \). Therefore,

\[
T(n) \geq T(u_0; n - |u_0|)
\]

\[
\geq \Theta(n/r) \Omega(r^2/\log r)
\]

\[
= \Omega\left(n/\log n\right) \left(\log^2 n / \log \log n\right)
\]

\[
= \Omega(n \log n / \log \log n). \quad \blacksquare
\]

**Inadequate Access to One-Dimensional Tapes**

On a higher-dimensional tape or a tree tape, any part of a ball can be reached and queried in time which is small compared to the volume of the ball. This allows an allegedly efficient simulator little time to revise its representation of the ball, so that a nearly static representation will have to suffice. This simplification leads to the nonlinear lower bounds derived above, but it does not yield any nontrivial lower bounds for simulation of one-dimensional tapes. Our general information-theoretic approach, however, can be used to give a simplified derivation of Aanderaa's result for one-dimensional tapes.

Our argument needs only a specialized version of Aanderaa's "Overlap Lemma". Our version, Lemma 4 below, deals with one particular sequence rather than with many, so no averaging is involved. Although our proof of Lemma 4 amounts to a specialization of Aanderaa's more general proof, we include it here for completeness.

**Definition [8, 18, 1].** Consider any sequence \( \ell_1, \ldots, \ell_n \) (*storage locations*). An overlap event in the subinterval \( I \subset [1, n] \) is a pair \((i, j)\) with \( i, j \in I \) and \( \ell_i = \ell_j \). Let \( W(I) \) be such a pair \((i, j)\) with \( i < j \) and \( \ell_i = \ell_j \). If \( \omega(I) \) is the number of overlap events \((i, j)\) in \( I \) with \( i \leq t < j \), then the *internal overlap* in \( I, \omega(I) \), is \( \max_{t \in I} \omega(I) \).

**Lemma 4.** For every \( N, r, s > 1 \) and every sequence \( \ell_1, \ldots, \ell_n \) of length \( r^sN \), there is some subinterval \( I \subset [1, r^sN] \) of length \( |I| = r^{s-1}N \) for some \( i \) (\( 1 \leq i \leq s \)) with \( \omega(I) \geq (1/s + 1/r)|I| \).

**Proof [1].** We use the simple fact that the length of each subinterval exceeds the number of overlap events in it. Parse the interval \([1, r^sN]\) into

- \( r \) subintervals \( I_{11}, I_{12}, \ldots \) of length \( r^{s-1}N \),
- \( r^2 \) subintervals \( I_{21}, I_{22}, \ldots \) of length \( r^{s-2}N \),

\[ \vdots \]

- \( r^s \) subintervals \( I_{s1}, I_{s2}, \ldots \) of length \( N \).

(Each parse is an \( r \)-fold refinement of the preceding one.) If the lemma fails, then the following holds for all \( i, j \):

\[
\omega(I_{ij}) > \left(1/s + 1/r\right)|I_{ij}| = (1/s + 1/r)r^{s-1}N.
\]

It follows that

\[
\sum_{i,j} \omega(I_{ij}) > \sum_{i=1}^{s} r^{i} (1/s + 1/r)r^{s-i}N
\]

\[
= (1 + s/r)r^{s}N.
\]
Below we obtain a contradictory upper bound.

For each $i, j$, select $t_{ij} \in I_{ij}$ so that $\omega_{t_{ij}}(I_{ij}) = \omega(I_{ij})$, thereby "distinguishing" the overlap events counted in $\omega_{t_{ij}}(I_{ij})$. By induction on $i$, select a subset $A$ of the subintervals $I_{ij}$:

$$I_{ij} \in A \leftrightarrow \text{there is no subinterval } I_{ij'} \in A \text{ with } t' < t \text{ and } I_{ij'} \in I_{ij}.$$

By design, all the distinguished overlap in the subintervals in $A$ is disjoint; so $\sum_{I_{ij} \in A} \omega(I_{ij}) < r^sN$. By induction on $i$, $A$ contains at most $r^{s-1}$ subintervals $I_{ij'}$ with $t' < t$; so $A$ omits at most $r^{s-1}$ of the $r^s$ subintervals $I_{ij}$. Noting that $\omega(I_{ij}) < |I_{ij}| = r^{s-1}N$, we conclude

$$\sum_{I_{ij} \in A} \omega(I_{ij}) < \sum_{i=1}^{r^{s-1}} r^{s-1}N = s/r \cdot r^{s-1}N.$$

Summing our two upper bounds, we get

$$\sum_{I_{ij}} \omega(I_{ij}) < r^sN + s/r \cdot r^{s-1}N = (1 + s/r)r^sN,$$

the desired contradiction. ■

Aanderaa's stated result is that fewer than $h$ single-head one-dimensional tapes cannot simulate $h$ such tapes, or even $h$ pushdown stores, in real time. His argument actually proves the slightly stronger assertion of our Theorem 3 below. Without loss of generality, that theorem will consider only real-time simulators with three convenient constraints:

- binary tape alphabets,
- separate commands for reads, writes, and shifts,
- exactly the same number of steps to handle each virtual command (call this number the delay of the simulator).

Theorem 3. Consider simulation of $h$ pushdown stores by single-head one-dimensional tapes. For each prospective real-time ($h - 1$)-tape simulator $M$ with delay $c$, there is a virtual command sequence $w$ on which $M$ errs. Moreover, $w$ need be no longer than some bound $N_{h,c}$ depending only on $h$ and $c$.

Proof: The idea is to push incompressible data at $h$ very different virtual rates, to find a virtual rate the prospective simulator neglects, and to use this neglect to get either an error or too short a description of the commands to the corresponding virtual store.

To make it clear which parameters below can depend on which others, we carefully order the assignment steps in our argument:

1. Consider delay-$c$ real-time simulation of $h$ pushdown stores.
2. Choose relative-push-density factor $d$ large enough for the analysis below. Set

$$S = \{ z_1z_2 \ldots z_h \mid z_i \text{ is a string of } d^i \text{ push commands to virtual store number } i \},$$

and let $\theta = d^2 + \cdots + d^h$ be the length of each string in $S$.
3. Choose "overlap fraction" $\varepsilon > 0$ small enough for the analysis below.
4. Choose $N$ large enough for the analysis below, and divisible by $ch\theta$ for convenience, intending to take $N_{h,c} = 2N^2 + 1$.
5. Consider any prospective real-time $(h - 1)$-tape simulator $M$ with delay $c$.
6. Take $z \in S$ with $K(z \mid M) \geq |z| = N^2\theta$. By this we mean, for some operational description $z \in \{0,1\}^*$ of $M$, to take $y \in \{0,1\}^*$ with $K(y \mid z) > |y| = N^2$, and then to let $z$ be that member of $S$ which pushes the sequence of bits $y$.

Assuming for the sake of argument that no prefix of $z$ is vulnerable to a sequence of subsequent pop commands to some virtual store, we will contradict the incompressibility of $z$. This will let us conclude that $M$ errs on some command sequence $w = yz$, where $y$ is a prefix of $z$ and $z$ is a sequence of pop commands to some virtual store, with $|z| \leq |y|$, so that $|yz| \leq 2|y| \leq 2|z| \leq 2N^2 < N_{h,c}$.

On virtual command sequence $z$, $M$ computes for $N^2$ steps. Consider the corresponding sequence $t_1, \ldots, t_N$ of most recently accessed storage locations. Choose a nonnull subinterval $I \subseteq [1, cN^2]$ with $|I|$ divisible by $N$ and $\omega(I) \leq e|I|$. By Lemma 4, this is possible if $cN \geq r^c$ for $r = e = 2/c$. (Recall that we are allowed to choose $N$ large in terms of $e$.) It is within time interval $I$ that we will consider the rates at which the prospective simulator’s heads move. Low overlap on the prospective simulator’s one-dimensional tapes will force the heads to go farther and farther from the data they have recorded.

Once we have $I$, we parse it into subintervals $I_{ij}$ of length $|I|/h$. In each of these subintervals, the number of commands to virtual store number $i$ is $n_i = (d^i/\theta)|I|/(ch\theta)$. Because all the data pushed onto virtual store number $i$ in $I$ can be retrieved in $n_i$, subsequent pop commands to that store, $M$ will have to be able to retrieve that same data without access to any tape square farther than $chn_i$ from the head positions at the end of $I$. Say that $M$ "neglects" virtual store number $i$ in $I$ if each head that ranges farther than $(ch\theta)n_i$ in $I$, ranges farther than $chn_i$ in the concatenation of the subsequent subintervals (call this concatenation $I_{i+1}$).
To see that \( M \) neglects some virtual store in \( I \), suppose it does not. For each \( i (1 \leq i \leq h) \), then, let \( j(i) \) be a head that ranges farther than \((ch/d)n_i/2\) in \( I_i \) but no farther than \( chn_i \) in \( I_{>2} \). One of \( M' \)'s \( h - 1 \) heads must serve as both \( j(i) \) and \( j(i') \) for \( i < i' \). But then that overworked head must range no farther than \( chn_i \) in \( I_{>2} \), yet farther than \((ch/d)n_{i+1} \), contradicting the induction hypothesis.

Assume that \( M \) neglects virtual store number \( i \) in \( I \). Because \( o(|I|) \leq |I| \), each head that visits more than \((ch/d)n_i \) tape squares in \( I_i \) revisits at most \(|I| \) of them in \( I_{>2} \) and ends up at least \( chn_i - 2e|I| \) tape squares away from them. (It is safe to assume \( chn_i > (ch/d)n_i = (d^{-1}/\theta)|I| \geq (1/\theta)|I| > 2e|I| \), since we are allowed to choose \( e \) small in terms of \( \theta \).) Even in the next \( chn_i \) steps by \( M \), therefore, no more than \( 2e|I| \) of the tape squares can be revisited. It follows that the \( n_i \) bits pushed onto virtual store number \( i \) in \( I_i \) can be recovered from \( M \), \( O(h((ch/d)n_i + e|I|)) \) bits of its instantaneous description at the end of \( I_i \), and the rest of \( x \). If we provide the sequence of bits pushed by the latter literally, along with \( i \), the location and length of \( I \), and an appropriate formalization of this whole discussion (much as in the proof of Lemma 1), then we get:

\[
K(x | M) \leq O(h((ch/d)n_i + e|I|))
\]

\[
+ (|x| - n_i) + O(|x|/|I|) + O(1 + h + c + d).
\]

substituting \( |I| = chn_i/d \leq chn_i/d \), we get

\[
K(x | M) \leq |x| + O(|x|/|I|) + O(1 + h + c + d) - n_i
\]

\[
+ O((1 + e\theta)ch^2n_i/d)
\]

\[
\leq |x| + O(|x|/|I|) + O(1 + h + c + d) - n_i/2,
\]

since we are allowed to choose \( e \) small in terms of \( \theta \), and \( d \) large in terms of \( c \) and \( h \). Substituting \( n_i = (d^2/\theta)|I|/(ch\theta) \geq dN/(ch\theta) \) = \( \sqrt{|x|/(ch\theta)} \), we finally get

\[
K(x | M) \leq |x| + O(|x|/|I|) + O(1 + h + c + d)
\]

\[
- d\sqrt{|x|/(2ch\theta)}
\]

\[
< |x|,
\]

since we are allowed to choose \( |x| = N^2 \) large in terms of \( c, h, d, \) and \( \theta \). ■

Corollary. Even an \((h - 1)\)-head tape with head-to-head jumps cannot simulate \( h \) pushdown stores in real time.

Proof: By induction on \( h' < h \), we prove that an \( h' \)-head tape with head-to-head jumps cannot simulate \( h \) pushdown stores in real time. The base case, \( h' = 1 \), follows trivially from Theorem 3. In the induction case, the idea is to drive the \( h' \) heads of any alleged simulator \( M \) with delay \( c \) farther apart than \( cN_{h,c} \). The next head-to-head jump at least \( N_{h,c} \) virtual commands \((cN_{h,c}) \) steps away, and then to cite Theorem 3 as if the \( h' \) heads were on separate tapes. Note that the initial tape contents within distance \( cN_{h,c} \) of each head can be managed in finite-state control, and that this does not change the delay \( c \).

In the only case omitted above, every virtual command sequence leaves a pair of \( M \)'s heads within distance \( cN_{h,c} \) or has an extension shorter than \( N_{h,c} \), which causes a head-to-head jump. But then there is a real-time machine \( M' \), described below, which correctly simulates the \( h \) pushdown stores if \( M \) does, and which has only \( h' = 1 \) heads. By the induction hypothesis \( M' \) errs, so \( M \) errs too.

We design \( M' \) to simulate \( M \) if \( M \) correctly simulates the \( h \) pushdown stores. While \( M \) has a pair of heads within distance \( c + cN_{h,c} \), \( M' \) stations a head at one of the positions and commits the relative position of the other to finite-state memory. In this case \( M' \) is able to handle the next virtual command just as \( M \) would, but with a longer (though still bounded) delay. When the pair of heads gets too far apart and some other pair of heads is within distance \( c + cN_{h,c} \), the machine just as \( M \) would, shift to its new post nearby. When the pair of heads gets too far apart and no other pair of heads is within distance \( c + cN_{h,c} \), there must be a sequence of at most \( N_{h,c} \) additional commands which causes a head-to-head jump by \( M \). Still within bounded delay, and temporarily suppressing all output, \( M' \) can find a shortest such sequence, simulate \( M \) on it, and then simulate \( M \) on an equally long sequence of commands which undoes the virtual damage, pushing what the first sequence popped and popping what it pushed. Before the repair, this leaves a pair of heads within distance \( c \). Even after the repair, therefore, it leaves a pair of heads within distance \( c + cN_{h,c} \), so that the simulation can continue. ■

Two-Tape Simulation of Multiple Tapes

The simulations we use and design in this section are particularly well-structured in the following sense: The simulators' tape heads return to the origins on their tapes at the completion of work on each input command. Call this sort of simulation \( H \)-simulation (“H” for “homing”). The value of \( H \)-simulation is that several \( H \)-simulations can interleave use of (different tracks on) the same tape units without interference or time loss. Although real-
time H-simulation is just finite-state transduction, there are nontrivial H-simulations which do not run in real time.

Lemma 5 [4, pp. 276-277]. A counter can be H-simulated in linear \(O(n)\) time by a single-head one-dimensional tape.

Corollary. Any fixed number of counters can be H-simulated in linear time by a single-head one-dimensional tape.

Lemma 6 [8]. A single-head one-dimensional tape can be H-simulated in time \(O(n \log n)\) by a pair of single-head one-dimensional tapes. (This H-simulation can also be "oblivious" [16].)

Corollary. Any fixed number of one-dimensional tapes, even with multiple heads [5,11] and head-to-head jumps [10], can be H-simulated in time \(O(n \log n)\) by a pair of single-head one-dimensional tapes.

Theorem 4. For \(d \geq 2\), any multithread \(d\)-dimensional tape, even with head-to-head jumps, can be H-simulated by a pair of single-head tapes, one \(d\)-dimensional and the other one-dimensional, in time \(O(n^{1+1/d-\alpha})\), where \(\alpha = 1/(d(d-1)+1))\).

Proof: To get a representation in small radius, the H-simulator strategy will be to pack nonblank "pages" of virtual storage compactly onto a "secondary storage" track of the \(d\)-dimensional tape. To get by efficiently with a single head on that tape, the strategy will be to copy "active" pages into the vicinity of the origin on a "primary storage" track of the same tape. The major problem will be to find the right pages in secondary storage fast enough. Our solution will involve bounded-depth recursion of the entire H-simulation.

Inductively, we will describe a sequence of H-simulation procedures, each successive one of which "recurses one level deeper". Each procedure will use a pair of single-head tapes, one \(d\)-dimensional and the other one-dimensional, to H-simulate the virtual \(d\)-dimensional tape. For each \(k\), there will be a constant \(c_k\) such that the time \(T_k(n)\) for H-simulation procedure \(k\) will satisfy

\[T_k(n) \leq c_k U_k(n)\]

\[U_k(n) = n^{1+1/d-\alpha} + (n^3)^k + \delta = 1-1/d^2\]

It will follow that \(T_k(n) = O(n^{1+1/d-\alpha})\) for any sufficiently large \(n\), as desired.

Let H-simulation procedure 0 be a naive one, with \(T_0(n) = O(n^3)\), say. (A pair of one-dimensional tapes suffices.) For \(k > 0\), the key to H-simulation procedure \(k\) will be a procedure \(SIM_k(n)\) to H-simulate the first \(n\) virtual commands in total time \(O(U_k(n))\) when \(n\) is provided as an auxiliary read-only off-line input (in unary notation, say). H-simulation procedure \(k\) will then be something like

\[\text{for } i = 1, 2, 3, \ldots \text{ do } [SIM_k(2^i); \text{ erase tapes}]\]

To provide the repetitious input this would require, and to properly screen the repetitious output it would generate, we include an input manager and an output manager. To make erasing easy, we modify \(SIM_k(2^i)\) to specially mark each square it visits. Erasure can then be achieved during a depth-first traversal of the connected graph formed by the marked squares on each tape; each marked tape square is erased the last time it is visited.

Through virtual command \(n\), the input and output managers and the loop control require only

\[O(2 + 2^2 + \ldots + 2^{\log_2 n}) = O(n)\] commands to a fixed finite set of one-dimensional tape units. These commands can be H-simulated by the Hennie-Stearns procedure (Lemma 6 and its corollary) in time \(O(n \log n)\) on tracks of the two tapes actually available. The time for \(SIM_k(2^i)\) and the subsequent erasure is \(O(U_k(2^i))\), so the total time to H-simulate the first \(n\) virtual commands (for every \(n\) now) is

\[O(n \log n + U_1(2) + U_2(2^2) + \ldots + U_k(2^{\log_2 n})) = O(U_k(n))\],

as desired.

It remains only to describe and analyze \(SIM_k(n)\). It is in this procedure that the main tracks on the \(d\)-dimensional tape will be one for "primary storage" and one for "secondary storage". Tracks of the one-dimensional tape will be used for (linear-time H-simulations of) counters (Lemma 5 and its corollary) and for assistance in copying "pages" of virtual storage between primary and secondary storage on the \(d\)-dimensional tape. In addition, on separate tracks of its tapes, \(SIM_k(n)\) will make use of the inductively available H-simulation procedure \(k-1\).

Each page of virtual storage will be a \(d\)-dimensional cube of \(b^d\) tape squares, where \(b = \lfloor \log_2 n \rfloor / (d(d-1)+1)\) = \(O(n^{1/(d(d-1)+1)})\). To H-simulate the first \(n\) virtual commands, \(SIM_k(n)\) will H-simulate \([n/b]\) time intervals, each one (except possibly the last) \(b\) virtual commands long. At the beginning of each such time interval, the \(3^d\) pages nearest each virtual head position will be found and loaded into primary storage, around the origin. The next \(b\) virtual commands can then be H-simulated directly in primary storage without any virtual head leaving the \(O(3^d)\) pages there. At the end of the time interval, the \(O(3^d)\) pages will be copied back to their locations in secondary storage. Those pages not yet assigned locations will be assigned vacant locations as close as possible to the origin in
Therefore, the total time for stored in binary, then each representative above has to from binary (O((n/b)(n/b)l/d b ) log(n/b)) simulator step counted below.

Note that some care is required to copy a page between primary and secondary storage. The page has to be copied onto and from the one-dimensional tape (in row-major order, say). Since both tape heads are involved, the linear-time counter H-simulations on the one-dimensional tape are not available to signal the ends of rows, etc. A simple solution is to use those counters ahead of time to prepare a "form" to copy onto and from on the one-dimensional tape. The following activities account for all H-simulator steps:

1. O(1) commands to counters for each other H-simulator step counted below.
2. Initial calculation of such constants as k, n/b, and log(n/b) (O(n) H-simulator steps).
3. O((n/b) log(n/b)) = O(n^{1-1/(d(d-1)+1)} log n) = O(n^d) commands to H-simulation procedure k - 1 (T_{k-1}(O(n^d)) H-simulator steps).
4. Conversion of page location components to and from binary (O((n/b)(n/b)^{1/d}) H-simulator steps).
5. Shifting to and from pages in secondary storage (O((n/b)(n/b)^{1/d}) H-simulator steps).
6. Copying between primary and secondary storage (O((n/b)b) H-simulator steps).
7. Direct H-simulation of virtual commands in primary storage (O((n/b)b^2) H-simulator steps).

Therefore, the total time for SIM_M(n) is

\[ O(T_{k-1}(O(n^d)) + (n/b)((n/b)^{1/d} + b^d)) \]

\[ = O((n^d)^{1+1/d-\alpha} + ((n/b)^{d-1} + n^{1+1/d-\alpha}) \]

\[ = O(U_k(n)), \]

as required. ■

Remaining Questions

Consider simulation of h single-head d-dimensional tapes by an h'-head d'-dimensional tape. Probably only for h' ≥ h and d' ≥ d is simulation possible in real time.

Hennie's result [7, 6] leaves open only cases with h' < h. We handle those cases with h' < h and d' < d^2 (Theorem 1 above), and Aanderaa handles those cases with h' < h and d' = d = 1 (Theorem 3 above). The remaining, still open cases have h' < h but d' ≥ max (d^2, 2).

Handling cases with h' < h and d' > d = 1 above might involve generalizing Aanderaa's proof (Theorem 3).

The problem is that low overlap seems less helpful on a higher-dimensional tape. It might help, however, to consider cases with h' much smaller than h, so that some simulator head has to handle many different virtual head rates. Perhaps it would help to obtain a time interval in which all simulator head motion is "essentially linear", with each overlap event spanning only a relatively short time interval.

To further pursue the question of what is needed to compensate for inadequate access into storage, one can consider even more exotic tapes (tree-shaped, for example) for the simulator heads, or even more restricted tapes for the virtual heads. For the proof of Theorem 3, the pushdown stores' virtual heads need only one "turn" (switch from pushing to popping, or vice versa) among them; but we do not know whether this is enough for the corollary. A more drastic restriction would replace the virtual pushdown stores by h counters. In this case, a one-head simulation is possible in linear time [4]; so the remaining question concerns real-time simulation. Note that the related "origin-crossing problem" is solvable in real time, by a one-head Turing machine [3].

Aanderaa's proof shows that the prospective simulator gets "caught off base" at some time when there is low overlap. Our intuition suggests that the prospective simulator can be caught off base at practically any point in the incompressible command sequence we use, and that the overlap lemma (Lemma 4 above) is not even needed. If the proof can thus be freed from reliance on that lemma, then it might become easier to adapt it for other purposes. For example, we might be able to consider virtual queues (first in, first out), or we might be able to show that simulation is not possible even in linear time. (Without the real-time assumption, there might
not happen to be enough virtual commands in the lowoverlap interval.)

Acknowledgement: This research was supported in part by the National Science Foundation under grants MCS77-06613 and MCS79-05006. It was facilitated by the use of Theory Net (NSF grant MCS78-01689). A preliminary report was presented on April 30, 1980 at the Twelfth Annual ACM Symposium on Theory of Computing, in Los Angeles, California.

References


PUBLICATIONS
1979-1980

Books and Chapters


Refereed and Conference Papers


Gacs, P., "Relation Between the Complexity and the Probability of Large Numbers," submitted to Theoretical Computer Science.


Peterson, G.L., "Concurrent Reading While Writing," TR67, Computer Science Department, University of Rochester, June 1980; submitted for publication.


Shani, U., "A Friendly Programming Environment for a Minicomputer," MIMI-79, Montreal, Quebec, September 1979; also TR54, Computer Science Department, University of Rochester, May 1980.


Sloan, K.R., Jr. and D. H. Ballard, "Experience with the Generalized Hough Transform," to be presented at the International Joint Conference on Pattern Recognition, Miami, Florida, December 1980; also, Proceedings, DARPA Image Understanding Workshop, College Park, Maryland, April 1980; also TR83, Computer Science Department, University of Rochester, October 1980.


Technical Reports and Unrefereed Papers


Gacs, P. and A. Yao, "Khachian's Algorithm Modified," TR81, Computer Science Department, University of Rochester, December 1979.


Lantz, K.A., "Uniform Interfaces for Distributed Systems," TR63, Computer Science Department, University of Rochester, July 1980.

Low, J.R., "Name-Type-Value (NTV) Protocol," TR73, Computer Science Department, University of Rochester, May 1980.


Rackoff, C.W. and J.J. Seiferas, "Limitations on Separating Nondeterministic Time Complexity Classes," TR80, Computer Science Department, University of Rochester, June 1980.

Rashid, R.F., "LIGHTS: A Study in Motion," Proceedings, DARPA Image Understanding Workshop, Los Angeles, California, November 1979; also, TR53, Computer Science Department, University of Rochester, May 1979.


Shopiro, J.E., "A Very High Level Language and Optimized Implementation Design for Relational Databases," TR66, Computer Science Department, University of Rochester, April 1980.


Sloan, K.R., Jr., "Analysis of 'Dot Product Space' Shape Descriptions," TR74, Computer Science Department, University of Rochester, June 1980.


Williams, G.J., "Program Checking," TR57, Computer Science Department, University of Rochester, September 1979.
SEMINARS

Fall 1979

Richard Greenblatt, A. I. Laboratory, M.I.T.
"The LISP Machine"

Robert B. Tilove, University of Rochester
"Unsettling Issues Concerning the Computational Complexity of Geometric Problems"

Geoffrey Hinton, Univ. of California, San Diego
"What are Visual Images"

F. Lockwood Morris, Syracuse University
"Automatic Assignment of Type Schemes to Programs"

David Reed, Lab for Computer Science, M.I.T.
"Implementing Atomic Actions on Decentralized Data"

Robert F. Sproull, Carnegie-Mellon University
"The 8 x 8 Display"

Charles Rieger, University of Maryland
"Word Experts in a Distributed Model of Parsing"

Howard Jay Siegel, Purdue University
"PASM: A Multimicrocomputer System for Image Processing and Pattern Recognition"

Mitchell P. Marcus, Bell Laboratories
"A Theory of Human Syntactic Recognition"

Mallory Selfridge, Yale University
"A Process Model of Language Acquisition"

Keith A. Lantz, University of Rochester
"Uniform Interfaces for Distributed Systems"

Peter Gacs: Joint Seminar Statistics, Math and Computer Science
"Khachian's Algorithm for Linear Programming"

Charles E. Molnar, Washington University
"A Current View of the Peripheral Auditory System"

Walter E. Daniels, IBM Watson Research Center
"A Programming System"

Jack C. Wileden, University of Massachusetts
"DREAM: A Support System for Designers of Distributed Computing Systems"

Spring 1980

Andrew Ortony, University of Illinois
"The Metaphor-Similarity Relation"

David J. DeWitt, University of Wisconsin
"Processor Allocation Strategies for Multiprocessor Database Machines"

Arnold L. Rosenberg, University of Toronto;
on leave from IBM Watson Research Center
"Issues in the Study of Data Encodings"

Mark J. Stefik, Stanford University
"Planning with Constraints"

Jonathan E. Shaprio, University of Rochester
"A Very High-Level Language and Optimized Implementation Design for Relational Databases"

Robert Paige, Rutgers University
"Plans to Implement Formal Differentiation -- A Technique for Improving Algorithms Specified at an Abstract Level"

Vaughn R. Pratt, M.I.T.
"Dynamic Algebra"

Carla S. Ellis, University of Oregon
"Concurrent Operations on B-Trees"

Fred B. Schneider, Cornell University
"Synchronization in Distributed Systems"

Leonid A. Levin, M.I.T.
"Complexity, Information and Foundations of Probability Theory"

Steven L. Small, University of Maryland
"Word Expert Parsing"

Marc H. Raibert, Calif. Institute of Technology.
"Dexterous Control of Arms and Legs: A Computational Problem"

Nancy A. Lynch, Georgia Institute of Technology
"A Model for Complexity Analysis of Distributed Systems"

Summer 1980

Robert F. Rosin, Bell Laboratories
"Communication System Architecture"

H. T. Kung, Carnegie-Mellon University
"Special-Purpose Chips: An Opportunity in VLSI"

David A. Patterson, Univ. of California, Berkeley
"Implications of VLSI in Future Architectures"

G. Reinhard Kofer, SIEMENS AG, Munich
"Userfriendly Program Development via Non-Procedural Tools"

Rohit Parikh, Boston University and M.I.T.
"A Survey of Propositional Dynamic Logics"

Takeo Kanade, Carnegie-Mellon University
"Mapping Image Properties into Shape Constraints for 3-D Shape Recovery from Images"
GRANT SUPPORT

Alfred P. Sloan Foundation: Computer Science
Department Support

Cook Engineering Company: Doctoral Level Research

Defense Advanced Research Projects Agency: "Knowledge-Directed Image Understanding in Photointerpretation"

Ford Motor Company: Doctoral Level Research

National Institutes of Health: "Anatomical Models in Computer-Aided Image Analysis"

National Science Foundation: "Advanced Compiling and Distributed Computing"

National Science Foundation: "Concurrency in the Access and Manipulation of Information Structures"

National Science Foundation: "Industry/University Cooperative Research Activity: PADL-2 Project"

National Science Foundation: "Recognizing Intent in Natural Language"

National Science Foundation: "Research in Manufacturing Modeling and Planning"

Sperry-Rand Corporation: Research on Programming Languages, Machine and Network Architecture, and Distributed Computing

Office of Naval Research: Center of Excellence in Artificial Intelligence: Knowledge Representation/Robotics

Xerox Corporation: Research Equipment Grant