Documents mean more than just Paper![1]

T. V. Raman
Cambridge Research Lab
Digital Equipment Corporation
Cambridge MA 02139
Phone: (617) 621-6637
E-mail: ⟨raman@crl.dec.com⟩

David Gries
Computer Science
Cornell University
Ithaca NY 14853
Phone: (607) 255-9207
E-mail: ⟨gries@cs.cornell.edu⟩

[1]A sequel to Documents are not just for Printing.

Abstract

With the advent of electronic documents, information is available in more than just its visual form —electronic information is display-independent. Though the principal mode of display is still visual, we can now produce alternative renderings of this information —we have designed a computing system, AsTeR, that produces an *audio view.*

The visual mode of communication is characterized by the spatial nature of the display and the eye's ability to actively access parts of this display. The reader is active, while the rendering itself is passive.

This active-passive role is reversed by the temporal nature of oral communication: the information flows actively past a passive listener. The passive nature of listening prohibits multiple views —it is impossible to first obtain a high-level view and then "look" at portions of the information in detail. These shortcomings become particularly severe when presenting complex mathematics orally.

We overcome these problems of oral communication by developing the notion of *audio formatting* —a process that renders information structure in a manner attuned to an auditory display. The *audio layout* present in the output conveys information structure. AsTeR is an interactive system —the ability to browse information structure and obtain multiple views makes the listener an active participant in oral communication.

The resulting *audio documents* achieve effective oral communication of structured information from a wide range of sources, including literary texts and highly technical documents containing complex mathematics.

Documents mean more than just Paper![†]

T. V. Raman and David Gries

June 27, 1994

1 Motivation

Documents encapsulate structured information. Visual formatting renders this structure on a two-dimensional display (paper or a video screen) using accepted conventions. The visual layout helps the reader recreate, internalize and browse the underlying structure. The ability to selectively access portions of the display, combined with the layout, enables multiple views. For example, a reader can first skim a document to obtain a high-level view and then read portions of it in detail.

The rendering is attuned to the visual mode of communication, which is characterized by the spatial nature of the display and the eye's ability to actively access parts of this display. The reader is active, while the rendering itself is passive.

This active-passive role is reversed in oral communication: information flows actively past a passive listener. This is particularly evident in traditional forms of reproducing audio, e.g., cassette tapes. Here, a listener can only browse the audio with respect to the underlying time-line —by rewinding or forwarding the tape. The passive nature of listening prohibits multiple views —it is impossible to first obtain a high-level view and then "look" at portions of the information in detail.

Traditionally, documents have been made available in audio by trained readers speaking the contents onto a cassette tape to produce "talking books". Being non-interactive, these do not permit browsing. They do have the advantage that the reader can interpret the information and convey a particular view of the structure to the listener. However, the listener is restricted to the single view present on the tape. In the early 80's, text-to-speech technology was combined with OCR (Optical Character Recognition) to produce "reading machines". In addition to being non-interactive, renderings produced from scanning visually formatted text convey very little structure. Thus, the true audio document was non-existent when we started our work.

We overcome these problems of oral communication by developing the notion of *audio formatting* —and a computing system that implements it. Audio formatting renders information structure orally, using speech augmented by non-speech sound cues. The renderings produced by this process are attuned to an auditory display —*audio layout* present in the output conveys information structure. Multiple audio views are enabled by making the

[†]A sequel to Documents are not just for Printing.

renderings interactive. A listener can change how specific information structures are rendered and browse them selectively. Thus, the listener becomes an active participant in oral communication.

In the past, information was available only in a visual form, and it required a human to recreate its inherent structure. Electronic information has opened a new world: Information can now be captured in a display-independent manner —using, e.g., languages like SGML[1] [Gol90] and (LA)TEX[2] [Lam86, Knu86, ?]. Though the principal mode of display is still visual, we can now produce alternative renderings, such as oral and tactile displays. We take advantage of this to audio-format information structure present in (LA)TEX documents. The resulting *audio documents* achieve effective oral communication of structured information from a wide range of sources, including literary texts and highly technical documents containing complex mathematics.

The results of this work are equally applicable to producing audio renderings of structured information from such diverse sources as information databases and electronic libraries. Audio formatting clients can be developed to allow seamless access to a variety of electronic information, available on both local and remote servers. Thus, the server provides the information, and various clients, such as visual or audio formatters, provide appropriate views of the information. Our work is therefore significant in the area of developing adaptive computer technologies.

Today's computer interfaces are like the silent movies of the past! As speech becomes a more integral part of human-computer interaction, our work will become more relevant in the general area of user-interface design, by adding audio as a new dimension to computer interfaces.

2 What is ASTER?

ASTER is a computing system[3] for producing audio renderings of electronic documents. The present implementation works with documents written in the TEX family of markup[4] languages: TEX, LATEX and $\mathcal{AMS}$-TEX. We were motivated by the need to render technical documents, and much effort was spent on designing audio renderings of complex mathematical formulae. However, ASTER works equally well on structured documents from non-technical subjects. Finally, the design of ASTER is not restricted to any single markup language —all that is needed to handle documents written in another markup language is a recognizer for it.

ASTER recognizes the logical structure of a document embodied in the markup and rep-

[1]Standard Generalized Markup Language (SGML) captures information in a layout independent form.

[2]LATEX, designed by Leslie Lamport, is a document preparation system based on the TEX typesetting system developed by Donald Knuth.

[3]In real life, ASTER is a guide-dog, a big friendly black Labrador.

[4]To most people, *markup* means an increase in the price of an article. Here, "markup" is a term from the publishing and printing business, where it means the instructions for the typesetter, written on a typescript or manuscript copy by an editor. Typesetting systems like (LA)TEX have these commands embedded in the electronic source. A *markup language* is a set of means (constructs) to express how text (i.e., that which is not markup) should be processed, or handled in other ways.

resents it internally. The internal representation is then rendered in audio by applying a collection of rendering rules written in AFL, our language for audio formatting. Rendering an internalized high-level representation enables AsTeR to produce different views of the information. A user can either listen to an entire document, or browse its internal structure and listen to portions selectively. The rendering and browsing components of AsTeR can also work with high-level representations from sources such as OCR-based document recognition.

This paper gives an overview of AsTeR, which is implemented in Lisp-CLOS with an Emacs front-end. The recommended way of using it is to run Lisp as a subprocess of Emacs. Throughout, we assume familiarity with basic Emacs concepts. For full details on the various components of AsTeR, see [Ram94]. See [Ram92, ?] for a description of some of our initial work in this area.

Section 3 introduces AsTeR by showing how documents can be rendered and browsed. Section 4 explains how AsTeR can be extended to render newly defined document structures in (LA)TEX. Section 5 gives some examples of changing between different ways of rendering the same information. Section 6 presents some advanced techniques that can be used when listening to complex documents such as text-books. AsTeR can render information produced by various sources. We give an example of this by demonstrating how AsTeR can be used to interact with the Emacs calculator, a full-fledged symbolic algebra system.

3 Rendering documents

This section assumes that AsTeR has been installed and initialized. At this point, text within any file being visited in Emacs (in general, text in any Emacs buffer) can be rendered in audio. To listen to a piece of text, mark it using standard Emacs commands and invoke read-aloud-region[5]. This results in the marked text being audio formatted using a standard rendering style. The text can constitute an entire document or book; it could also be a short paragraph or a single equation from a document —AsTeR renders both partial and complete documents. This is the simplest and also the most common type of interaction with AsTeR.

The input may be plain ASCII text; in this case, AsTeR will recognize the minimal document structure present —e.g., paragraph breaks and quoted text. On the other hand, (LA)TEX markup helps AsTeR recognize more of the logical structure and, as a consequence, produce more sophisticated renderings.

Browsing the document

Next to getting AsTeR to speak, the most important thing is to get her to stop speaking. Audio renderings can be interrupted by executing reader-quit-reading[6]. The listener can then traverse the internal structure by moving the *current selection*, which represents the current position in the document (e.g., current paragraph), by executing any of the browser commands reader-move-previous, reader-move-next, reader-move-up or reader-move-down.

[5]This is an Emacs Lisp command, and in the author's setup, it is bound to C-z d.

[6]Bound to C-b q.

To orient the user within the document structure, the current selection is summarized by verbalizing a short message of the form "<*context*> is <*type*>", e.g., moving down one level from the top of the equation

$$\sum_{1<=i<=n} i = \frac{n(n+1)}{2} \tag{1}$$

AsTeR speaks the message "*left hand side* is *summation*". The user has the option of either listening to just the current selection (reader-read-current) or listening to the rest of the document (reader-read-rest).

Examples of use

AsTeR can be used to:

- Read technical articles and books. The files for such documents may be available on the local system or on the global Internet[7]. Resources retrieved over the network can be audio formatted by AsTeR, since they are just text in Emacs buffers. The author has listened to this thesis as well as 10 textbooks using AsTeR. In addition, AsTeR has rendered a wide collection of technical documents available on the INTERNET, including technical reports and **AMS** bulletins.

- Entertain. About 200 electronic texts are available on the INTERNET, including the complete works of Shakespeare. The majority of these documents are in plain ASCII, but the quality of audio renderings produced by AsTeR, based on the minimal document structure that can be recognized, still surpasses the output of conventional reading machines. Increased availability of electronic texts marked up in (LA)TeX and SGML will enable better recognition of document structure and, as a consequence, better audio renderings.

- Proof-read partial and complete documents under preparation. This feature is specially useful when typesetting complex mathematical formulae. This paper has been proof-read using AsTeR and the system helped the author locate several minor errors, including bad punctuation. Thus, though designed as a system for rendering documents, the flexible design, combined with the power afforded by the Emacs editor, turns AsTeR into a very useful document preparation aid.

4 Extending AsTeR

The quality of audio renderings produced by AsTeR depends on how much of the document logical structure is recognized. Authors of (LA)TeX documents often use their own macros[8] to encapsulate specific logical structures. Of course, AsTeR does not initially know of these

[7] ANGE-FTP, an Emacs utility written by Andy Norman, allows seamless access to remote files. In addition, Emacs clients are available for networked information retrieval systems like GOPHER, WWW and WAIS.

[8] Macros permit an author to define new language constructs in TeX and specify how these constructs should be rendered on paper.

extensions. User-defined (LA)TEX macros are initially rendered in a canonical way; typically, they are spoken as they appear in the running text. Thus, given a document containing

```
$A \kronecker B$
```

ASTER would say

> cap a kronecker cap b

In this case, this canonical rendering is quite acceptable.

In general, how ASTER renders such user-defined structures is fully customizable. The first step is to extend the recognizer to handle the new construct, in this case, `\kronecker`.

The recognizer is extended by calling Lisp macro define-text-object as follows:

```
(define-text-object :macro-name "kronecker" :number-args 0
:processing-function kronecker-expand
:object-name kronecker
:supers (binary-operator) :precedence  multiplication)
```

This extends the recognizer; instances of macro `\kronecker` are represented by object *kronecker*. The user can now define any number of renderings for instances of object *kronecker*.

AFL, our language for audio formatting, is used to define rendering rules. Here is one such rendering rule for object *kronecker*:

```
(def-reading-rule (kronecker simple)
    "Simple rendering rule for object kronecker."
  (read-aloud "kronecker product of ")
  (read-aloud (first (children kronecker)))
  (read-aloud " and ")
  (read-aloud (second (children kronecker))))
```

ASTER would now speak `$A \kronecker B$` as

> kronecker product of cap a and cab b.

Notice that the order in which the elements of $A \otimes B$ are spoken is independent of the order in which they appear on paper. ASTER derives its power from representing document content as objects and by allowing multiple user-defined rendering rules for individual object types. These rules can cause any number of audio events (ranging from speaking a simple phrase, to playing a digitized sound). Once the recognizer has been extended by an appropriate call to define-text-object, user-defined macros in (LA)TEX can be handled just as well as any standard (LA)TEX construct.

To give an example of this, the logo that appears on the first page of the first author's PhD thesis is produced by (LA)TEX macro `\asterlogo`. After extending the recognizer with an appropriate call to define-text-object, we can define an audio rendering rule that produces a bark when rendering instances of this macro.

5 Producing different audio views

AsTeR can render a given object in more than one way. The listener can switch among any of several predefined renderings for a given object to produce different *views*, or add to these by defining new rendering rules.

Activating a rendering rule is the simplest way of changing how a given object is rendered. Statement

(activate-rule <*object-name*> <*rule-name*>)

activates rule <*rule-name*> for object <*object-name*>. Thus, executing

(activate-rule 'paragraph 'summarize)

activates rule summarize for object *paragraph*.

Suppose we wish to skip all instances of *verbatim* text in a LaTeX document. We could define and activate the following *quiet* rendering rule for object *verbatim*:

```
(def-reading-rule (verbatim quiet) nil)
```

Later, to hear the *verbatim* text in a document, the previously activated rule *quiet* can be deactivated by executing

```
(deactivate-rule 'verbatim)
```

Notice that at any given time, only one rendering rule is active for any object. Hence, we need only specify the object name when deactivating a rendering rule.

Activating a new rule is a convenient way of changing how instances of a specific object are rendered. Rendering *styles* enable the user to make more global changes to the renderings. Activating style style-1 by executing

```
(activate-style 'style-1)
```

activates rendering rule style-1 for all objects for which this rendering rule is defined. All other objects continue to be rendered as before. This is also true when a sequence of rendering styles is successively activated. Thus, activating rendering styles is a convenient way of progressively customizing the rendering of a complex document.

The effect of activating a style can be undone at any time by executing

(deactivate-style <*style-name*>)

AsTeR provides the following rendering styles:

- Variable-substitution: Use variable substitution to render complex mathematical expressions.

- Use-special-pattern: Recognize special patterns in mathematical expressions to produce context-specific renderings. For example, this enables AsTeR to speak A^T as "cap a transpose".

- Descriptive: Produce descriptive renderings for mathematical expressions.
- Simple: Produce a base-level audio notation for mathematical expressions.
- Default: Produce default renderings.
- Summarize: Provide a summary.
- Quiet: Skip objects.

When AsTeR is initialized, the following styles are active, with the leftmost style being the most recently activated style.

```
(use-special-pattern descriptive simple default)
```

Defining a new rendering style is equivalent to defining a collection of rendering rules having the same name. Note that a rendering style need not provide rules for all objects in the document logical structure. As explained earlier, activating a style only affects the renderings of those objects for which the style provides a rule.

6 Using the full power of AsTeR

This section demonstrates some advanced features of AsTeR that are useful when rendering complex documents. AsTeR recognizes cross-references and allows the listener to traverse these as hypertext links. Cross-referenceable objects can be labeled interactively, and these labels can be used when referring to such objects within renderings. The ability to switch among rendering rules enables multiple views and allows the listener to quickly locate portions of interest in a document. By activating rendering rules, all instances of a particular object can be floated to the end of the containing hierarchical unit, e.g., all footnotes can be floated to the end of a paragraph. This is convenient when getting a quick overview of a document. AsTeR also provides a simple bookmark facility for marking positions of interest to be returned to later. Finally, AsTeR can be interfaced with sources of structured information other than electronic documents. Later, we demonstrate this by interfacing AsTeR to the Emacs calculator.

Cross-references

Cross-reference tags that occur in the body of a document are represented internally as instances of object *cross-reference* and contain a link to the object being referenced. Of course, how such cross-reference tags are rendered depends on the currently active rule for object *cross-reference*. The default rendering rule for cross-references presents the user with a summary of the object being cross-referenced, e.g., the number and title of a sectional unit. This is followed by a non-speech audio prompt. Pressing a key at this prompt results in the entire cross-referenced object being rendered at this point —rendering continues if no key is pressed within a certain time interval. In addition, the listener can interrupt the rendering and move through the cross-reference tags. This is useful in cases where many such tags occur within the same sentence.

Labeling a cross-referenceable object

Consider a proof that reads:

> By theorem 2.1 and lemma 3.5 we get equation 8 and hence the result.

If the above looks abstruse in print, it sounds meaningless in audio. This is a serious drawback when listening to mathematical books on cassette, where it is practically impossible to locate the cross-reference. ASTER is more effective, since these cross-reference links can be traversed, but traversing each link while listening to a complex proof can be distracting.

Typically, we only glance back at cross-references to get sufficient information to recognize theorem 2.1. ASTER provides a convenient mechanism for building in such information into the renderings. When rendering a cross-referenceable object such as an equation, ASTER verbalizes an automatically generated label (e.g., the equation number) and then generates an audible prompt. By pressing a key at this prompt, a more meaningful label can be specified, which will be used in preference to the system-generated label when rendering cross-references.

To continue the current example, when listening to theorem 2.1, suppose the user specifies the label "Fermat's theorem". Then the proof shown earlier would be spoken as:

> By Fermat's theorem and lemma 3.5 we get equation 8 and hence the result.

Of course, the user could have specified labels for the other cross-referenced objects as well, in which case the rendering produced almost obviates the need to look back at the cross-references.

Locating portions of interest

Printed books allow the reader to skim the text and quickly locate portions of interest. Experienced readers use several different techniques to achieve this. One of these is to locate an equation or table and then read the text surrounding it. ASTER provides this functionality to some extent.

We explained in Section 5 that different rules can be activated to change the type of renderings produced. Using this mechanism, we can activate a rendering rule (see Figure 1 on page 10) that speaks only the equations of a document. When a specific equation is located, rendering can be interrupted and a different rule activated. Using the browser, the listener can now move the current selection to the enclosing hierarchical unit (e.g., the containing paragraph or section) and listen to the surrounding text.

Getting an overview of a document

Rendering rules can be activated to obtain different views of a document. For instance, activating rendering rule *quiet* for an object is a convenient way of temporarily skipping over all occurrences of that object —activating *quiet* for object *paragraph* provides a thumb-nail view of a document by skipping all content. This is similar to skipping complex material when first reading a printed document.

```
(def-reading-rule (paragraph read-only-display-math)
    "Render only  the math appearing in paragraphs"
  (let ((math-in-paragraph
         (mapcar #'(lambda(object)
                      (when  (display-math-p object) object))
                  (contents paragraph ))))
    (mapc #'read-aloud  math-in-paragraph)))
```

Figure 1: Rendering only displayed math.

We may skip instances of some objects entirely, e.g., source code; in other cases, we may merely defer the reading. This notion of delaying the rendering of an object is aptly captured by the concept of *floating* an object to the end of the enclosing unit. Typesetting systems like (LA)TEX permit the author to float all figures and tables to the end of the containing section or chapter. However, only specific objects can be floated, and this is exclusively under the control of the author, not the reader of the document.

ASTER provides a much more general framework for floating objects. Any object can be floated to the end of any enclosing hierarchical unit —instances of object *footnote* can be floated to the end of the containing paragraph. The ability to float objects is useful when producing audio renderings, since audio takes time, and delaying the rendering of some objects provides an overview.

Rendering using variable substitution

When reading complex mathematics in print, we first get a high-level view of an equation and then study its various subexpressions. For example, when presented with a complex equation, an experienced reader of mathematics might view it as an equation with a double summation on the left-hand side and a double integral on the right-hand side, and only then attempt to read the equation in full detail. In a linear audio rendering, the temporal nature of audio prevents a listener from getting such a high-level view. We compensate by providing a variable-substitution rendering style. When it is active, ASTER replaces sub-expressions in complex mathematics with meaningful phrases. Having thus provided a top-level view, ASTER then renders these sub-expressions.

Bookmarks

The browser provides a simple bookmark facility for marking positions of interest to be returned to later. Browser command mark-read-pointer prompts for a bookmark name and marks the current selection. Later, the listener can move to the object at this marked position by executing browser command follow-bookmark with the appropriate bookmark name.

Interfacing AsTeR with other information sources

AsTeR has been presented as a system for rendering documents in audio. More generally, AsTeR is a system for speaking structured information. This fact is amply demonstrated by the following example, where we interface AsTeR to the Emacs calculator.

The Emacs calculator, a public domain symbolic algebra system, provides an excellent source of examples for trying out the variable-substitution rendering style. Creating such an audio interface could be challenging, since the expressions produced are quite complex. However, the flexible design of AsTeR and the power of Emacs makes this interface easy. A collection of Emacs Lisp functions encodes the calculator output in LaTeX and places it in an Emacs buffer, which AsTeR then renders.

A user of the Emacs calculator can execute command read-previous-calc-answer to have the output rendered by AsTeR. The expression can be browsed, summarized, transformed by applying variable substitution, and rendered in any of the ways described in the context of documents.

References

[Gol90] Charles F. Goldfarb. *The SGML handbook.* Oxford: Clarendon Press; Oxford; New York: Oxford University Press, 1990.

[Knu86] Donald E. Knuth. *TeX The Program.* Addison-Wesley, Reading, Mass., 1986.

[Lam86] Leslie Lamport. LaTeX*: A Document Preparation System.* Addison-Wesley, Reading, Mass., 1986.

[Ram92] T. V. Raman. An audio view of (La)TeXdocuments. *Proceedings of the TeX Users Group*, 13:372–379, July 1992.

[Ram94] T. V. Raman. *Audio System for Technical Readings.* PhD thesis, Cornell University, May 1994.