METHODS & DESIGNS

Computerized transcripts of social interactions

WILLIAM GARDNER

University of Pittsburgh School of Medicine, Pittsburgh, Pennsylvania

The advantages of a standardized format for transcribing social interactions to computerized media are discussed. The chief advantage of this scheme is that transcripts can be easily exchanged among research groups and across text processing programs. An important element of a transcript is the set of conventions, called *markup*, that identify the metalinguistic features of texts. Conventions should employ symbols that (1) cannot be mistaken for ordinary text, (2) explicitly describe their linguistic function, and (3) obey a grammar. Social scientists who transcribe social interactions should participate in the development of a standardized scheme of descriptive markup for the encoding of machine-readable transcripts, based on the Standard Generalized Markup Language.

Transcribed social interactions are a data source in many of the social sciences. For example, interactions between therapists and clients are transcribed and analyzed in the training of psychotherapists and social workers, and they are employed in research by sociologists, psychologists, educators, linguists, and anthropologists (e.g., Bakeman & Gottman, 1986; Ericsson & Simon, 1984; Mishler, 1986; Schegloff, 1987). In this article, I discuss some principles relevant to encoding and exchanging transcribed social interactions in machine-readable files. The proposed scheme for computerized transcripts is designed to satisfy two goals. First, transcripts should be easily exchangeable among research groups and easily transferable across text processing programs. Second, it should be possible to integrate a transcript into a multimedia or hypertext document. To fulfill these goals, researchers using social interaction data should consider the adoption of a standard encoding scheme for machine-readable transcripts, based on the Standard Generalized Markup Language (for a proposed standard representation of sequential interaction data, see Bakeman & Quera, 1992; and for discussions of software tools to aid protocol analysis, see James & Sanderson, 1991; Tesch, 1990; Tesch, 1991a, 1991b).

Exchangeable Transcripts

At present, exchanging transcripts is inconvenient, which impedes good science. The computerizing of social interaction data should make interpretive analyses of

interactions more public and less idiosyncratic (Conrad & Reinhartz, 1984) by providing access to the transcribed data. If data sets moved freely across laboratories, investigators' claims could be more stringently reviewed by peers. Diverse data sets pertinent to a common topic could also be pooled for a large analysis.

Exchanging collections of printed transcripts is inconvenient, because they are difficult to browse and search. But exchanging computerized transcripts is also difficult. On the one hand, the transcript files may be encoded in the proprietary format of the investigator's word or text processing software. If so, documents created by that program are likely to remain tied to it. On the other hand, if the transcripts are encoded as plain ASCII text, they will be searchable only through general-purpose text editing tools. Browsing and searching with such primitive tools can be almost as difficult as reading the print. It is not so surprising that transcribed data sets rarely circulate.

Indeed, the problem is not simply that data sets are not exchangeable across laboratories, it is frequently a challenge just to move a data set across document processing applications within a laboratory.¹ Many document processing programs have filters that can, up to a point, translate foreign formats. But these filters are, shall we say, not wholly reliable, even when one is translating a document file across different versions of the *same* document processing application.² The solution to these problems is a rich but standardized format for encoding transcribed interactions, so that any program that can process the format can process a compliant document.

Multimedia Data Structures

When a transcript is generated from a (video or audio) recording of an interaction, there will be at least two channels of data available to a researcher. These records, however, are stored on different media, however, making the various records of the same interaction difficult to syn-

Although I am not a member of the TEI, I have learned greatly from their efforts. I have also benefited from monitoring discussions in the USENET news groups comp.text and comp.text.sgml, and the electronic TEI discussion group. I extend my thanks to the editor, Roger Bakeman, Charles Lidz, Thomas Muhr, Donald Spaeth, and an anonymous reviewer for comments on the manuscript. Address correspondence to W. Gardner, Law and Psychiatry Research, Department of Psychiatry, University of Pittsburgh School of Medicine, Pittsburgh, PA 15213 (e-mail: wpg@ethics.med.pitt.edu).

chronize. Multimedia documents that would allow browsable and searchable access to several synchronized channels of data on an interaction could be powerful tools for social interaction research. For example, there is considerable interest in the psychophysiology of emotion in social interactions (Levenson & Gottman, 1983, 1985). One can readily envision a workstation that could display a video record of, say, a couple's discussion of marital problems, synchronized with a transcript of the dialogue and plots of each partner's heart rate (MacKay & Davenport, 1989). What is needed, however, is a standard means to include time markers in textual transcripts, so that the transcript can be synchronized with the other data channels.

COMPUTERIZING TRANSCRIPTS

To define a standard set of transcript conventions we need a language for representing the information in the transcript. This article outlines a scheme for formatting computerized transcripts that is based on the Standard Generalized Markup Language (International Organization for Standardization [IOS], 1986). We begin with a discussion of the problems of transcripts and how they are addressed by SGML.

Problems in Computerizing Transcripts

In many ways, transcripts resist computerization. Among the features of transcripts that make them difficult to computerize are that they use human-oriented rather than computer-oriented notation and that they have variable rather than fixed discourse structures.

Markup in natural languages. Transcribed information is encoded in natural languages, and, in particular, with the standard markup of these languages. Markup is information in a text that tells the reader how to interpret other components of text. Typical markup resources include white space, fonts, capitalization, and punctuation. You can probably read this, but it is easier if I explicitly mark individual words by separating them with white space.

Transcripts utilize many markup conventions, because transcribing involves more than simply copying the words on a tape. One must also include information about who is speaking, and, frequently, about the speaker's affect, concurrent behavior, and so on. Because a transcript contains many kinds of text, we need conventions for signaling whether a segment should be interpreted as, for example, a subject's speech or a coding by an analyst.

Both human and computer readers need markup, but they like it written differently. For example, in the expressions $\sin(x)$ and $f(x)$, the parentheses help the reader see that x is the argument of the functions. But a human reader would not be confused if parentheses were used inconsistently, as in $\sin x$ and $f(x)$, because context, common usage, and the fonts still identify $\sin$ as a function and x as an argument. Humans prefer markup systems that include multiply defined but visually economical glyphs like "(", and they take the necessarily context-dependent rules for the usage of such marks to be a virtue. Computers are indifferent to clutter and prefer simple,

rigid markup, with consistent punctuation, no manipulation of fonts, and minimal dependence on context.

Variable structure. Transcripts have heterogeneous structures. For example, a transcript typically begins with a header that indicates the date of an interview, names or codes for the participants, and so forth. The body of the transcript follows, containing a number of speaking turns. However, the number of turns is not fixed, and other material, such as a description of the speaker's affect or narrative commentary by an observer, can also appear at unpredictable places. Most of these distinctions among kinds of text within a transcript are unknown to the computer, so that search and retrieval functions cannot be finely tuned to the researcher's goals. Similarly, it is difficult to mechanize the manipulation of transcripts to facilitate the generation of new documents from the components of existing records.

Machine-readable encoding of computerized transcripts. The problem is that although transcripts have intricate structures, the markup identifying those structures can be grasped only by human readers, not computers. If so, computerized transcripts are little more than blobs of typography, rendered in digital codes instead of paper and ink. The remedy is to shift from the use of natural language markup in computer-readable transcripts to a more explicit, less context-dependent style of computer-oriented markup, standardized across research groups. This markup should enable the computer to recognize distinctions among various text components in a transcript and to process the document accordingly—including translating the computer-encoded transcript into formatting suitable for human readers.

Standards for Machine-Readable Texts

The Text Encoding Initiative. The format for machine-readable encoding of social interactions discussed here is based loosely on guidelines for the computerization of academic text proposed by the Text Encoding Initiative (Barnard, Fraser, & Logan, 1988a; Text Encoding Initiative [TEI], 1990). The problems of how to computerize transcribed interactions overlap those faced by workers in the humanities who need to computerize literary and linguistic texts. The TEI is a consortium of the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing. There are also representatives from many other humanities groups, as well as the American Anthropological Association and the Information Retrieval Special Interest Group of the Association for Computing Machinery.

The TEI is developing a set of guidelines for formatting machine-readable texts (TEI, 1990). These guidelines are intended to meet many goals, including the following. First, the standard should specify a common interchange format, but through a set of guidelines, not a set of rigid requirements. Second, it must be sufficiently flexible and extensible to represent the textual markup that is currently used for both ordinary communication and textual research. Third, the standard should be hardware

and software independent, so that it can be used with any computer, program, storage device, or operating system. It should also be application independent—for example, it should not be suitable only for typesetting software or word processors. Fourth, it should be suitable for marking up texts in any human language. Finally, the TEI intended that the guidelines should provide a set of recommendations for encoding new textual materials, and hence this proposal to extend them to the encoding of interactional transcripts.

THE STANDARD GENERALIZED MARKUP LANGUAGE

The TEI guidelines for markup conform to the SGML, an internationally standardized metalanguage for developing machine-readable markup (IOS, 1986). Originating from a language developed at IBM (Bryan, 1988), SGML appears to be gaining considerable acceptance as a standard. The international standardization of SGML means that texts encoded following rules specified in its terms should be both portable and stable. SGML is not a set of markup conventions, it is a language in which such conventions can be defined.

Presentational, Procedural, and Descriptive Markup

There are several kinds of markup (see Coombs, Renear, & DeRose, 1987, for a lucid discussion of the distinctions that follow, and also Lampton, 1986, chap. 1). Presentational markup includes all the typographical devices used on a page (or other image) to mark up text for human readers. For example, a common item of presentational markup in transcribing social interactions is the convention that a change of speakers during a dialogue should be marked with a line skip. Text formatting code or word processor files will also contain procedural markup. Instead of including a blank line in the text file, which would be presentational markup, one includes an instruction to the computer that will produce the same effect (e.g., the command `.sk`, which produces a line skip in the UNIX *troff* formatter; an “invisible” embedded code for a hard carriage return produces the same effect in WordPerfect). A third kind of markup is descriptive markup. Instead of marking up a file as it would be presented to a human reader or including a formatting procedure for a computer, one inserts a token that explicitly states the discourse function of a segment of text. For example, you could mark the beginning of a turn with something like `<turn>`. The guillemets make the metatextual character of `<turn>` unmistakable, and the word makes its function self-explanatory.

Suppose, then, that we wished to mark up a transcript to identify speaking turns, the speakers, the times since the beginning of the interview, and the words spoken. We might print a transcript marked up as in the following example of natural language (presentational) markup:

Interviewer: (5:38) Could you tell me what has been going on in your life recently that has led you to coming here?

Patient: (5:46) I have been having these confusing thoughts and depression and violent thoughts in particular, that had become constant, so constant that I just can't even really function.

This printed document might have been generated from a WordPerfect file. If one viewed the file and gave the command to reveal the hidden formatting codes, it might look like the following transcript with procedural markup:

Interviewer: (5:38) Could you tell me what has been going on in your[SRt]
[Tab]life recently that has led you to coming here?[HRT]
[HRT]
Patient: (5:46) I have been having these confusing thoughts and[SRt]
[Tab]depression and violent thoughts in particular, that had[SRt]
[Tab]become constant, so constant that I just can't even[SRt]
[Tab]really function.[HRT]

The [HRT]s are hard carriage returns where the author has forced a linebreak, the [SRt]s are soft returns where the program has elected to break the line, and the [Tab]s create the indenting effect. These characters are examples of the procedural markup that tells the document processing application to do something (“break the line here” or “skip to the next tab stop”) when it renders the document on screen or paper. There are usually many procedural markup schemes that are rendered identically when a document is printed. For example, in WordPerfect 5, the preceding marked up text and the one below would be printed identically:

Interviewer: (5:38) Could you tell me what has been going on in your[SRt]
[Tab]life recently that has led you to coming here?[HRT]
[HRT]
Patient: (5:46) I have been having these confusing thoughts and[SRt]
[->Indent]depression and violent thoughts in particular, that had[SRt]
become constant, so constant that I just can't even really function.[HRT]

This example shows how different procedural markup schemes can map to the same presentational markup (i.e., the layout and typography of the printed document). Because it is likely that different transcribers will use different procedural markup (e.g., [Tab]s vs. [->Indent]s) to achieve the same presentational effect, it is difficult to create consistent computer-readable transcripts using procedural markup.

Descriptive Markup in SGML: Tags

SGML markup consists of pairs of *tags* that enclose (and, thus, identify) *elements* within a document. Elements may be composed of other elements, or of the primary text being marked up. For example, we could define a transcript as a document reporting one or more interviews between a doctor and patient. Thus, interviews would be elements of a transcript, and speaking turns, elements nested within interviews. But it could be encoded with SGML tags for the procedural markup, as is shown below. Here, the SGML markup is entirely in the tags. The white space is irrelevant to an SGML parser, which would interpret the text in the same manner as it would the second example below, which has the SGML tags marking up a continuous text stream.

```

<turn>
  <speaker>Interviewer</speaker>
  <time>5:38</time>
  <speech>Could you tell me what has been going on in your life recently that has led you to coming here?</speech>
</turn>
<turn>
  <speaker>Patient</speaker>
  <time>5:46</time>
  <speech>I have been having these confusing thoughts and depression and violent thoughts in particular, that had become
    constant, so constant that I just can't even really function.</speech>
</turn>

<turn><speaker>Interviewer</speaker><time>5:38</time><speech>Could you tell me what has been going on in your life re-
cently that has led you to coming here? </speech></turn><turn><speaker>Patient</speaker>
<time>5:46</time><speech>I have been having these confusing thoughts and depression and violent thoughts in particular,
that had become constant, so constant that I just can't even really function.</speech></turn>

```

Advantages of Descriptive Markup

The tags in the preceding examples illustrate two immediate advantages of descriptive markup. First, the descriptive markup prevents any ambiguity about the structure of the discourse by directly specifying the function of the enclosed segment of text. Second, it can be used to create documents that can be flexibly reformatted to suit the purpose at hand, because it encapsulates a logical structure without committing one to any specific presentational scheme.

Descriptive markup disambiguates the discourse structure. In presentational markup, every element of white space and punctuation has numerous discourse functions, which visually oriented humans resolve effortlessly in context, but which trouble computers. For example, colons are used to mark boundaries between both speaker and speech and between clauses within a sentence. How can we be sure that the computer will never try to read something like “is:” as marking the identity of a speaker named *is*? One could program the computer to recognize that the markup for a speaker is not just a colon but a line space, a capitalized word, and a colon. Would this knowledge of presentational markup enable the computer to pick out speaker identifiers every time and only when they occur? Perhaps, but in trying to capture complex, context-dependent markup we might place unwanted constraints on our text encoding. For instance, if we later wish to have “Second patient” be a speaker, we may need to reprogram the computer to recognize two-word speaker identifiers. Because the original system was not planned with this requirement in mind, such a change is virtually guaranteed to produce pernicious bugs.

Falling back, we might decide to retain one-word patient identifiers and require transcribers to write *Second _ Patient*. But for this rule to work, we need to make the computer enforce it on transcribers, because otherwise the underscore will sometimes be omitted. We could write a text editor macro prescribing a template for speeches that allowed only a one-word speaker name. But if we are already committed to a structured data entry format, why not use the macro to implement a descriptive-markup scheme? If speaker identifiers were enclosed within `<speaker> . . . </speaker>` tags, it would allow us to use whatever name seems most natural, without restrictions on number of words, capitalization, and so forth.

Descriptive markup is more informative. Descriptive markup provides sufficient information to specify the procedural or presentational markup of a text, but not vice versa. That is, one can take a file marked up descriptively and translate it into a file marked up procedurally, by substituting, for example, `.sk` for every `<turn>`. One can then process the procedural file to produce an image or print a document with presentational markup. Because visually identical markup may have diverse discourse functions, one cannot transform a presentational or procedural file into a descriptively marked up file without adding information to it. Descriptive markup is therefore ideal for the interchange of documents, because it contains the information necessary to produce presentational markup, but it does not commit one to any specific presentational scheme. Researchers can exchange texts with descriptive markup and then render the text on the page in a style suitable to their particular goals and research groups' standards. In summary, presentational and procedural markup are ambiguous, fragile, and difficult to maintain. Descriptive markup is unmistakable, robust, and straightforward to maintain.

Markup Grammar

Although it would be convenient for researchers to employ similar names to tag similar segments of text, the

Figure 1. Components of a transcript.

names of the tags are inessential. The computer, after all, has no idea what "turn" means. Similarly, one could substitute "{" and "}" for "<" and ">". What matters are the rules for a tag's usage. SGML enables and requires one to define a grammar for the tags (i.e., the markup) pertinent to a specific document type. As a result, the usage of the tags in an SGML document can be checked for syntactic correctness.

Documents and their grammars. Like other scientific documents, a transcript is organized in a stereotyped pattern, the component parts of which are explicitly marked up. This structure can be described as a tree. For example, Figure 1 depicts a transcript as a tree with leaves that include a header and a body. The header is another tree, whose detail is suppressed in the figure, but which would typically contain a date, the name of the transcriber, the subject of the interview, and so forth. The body is also a tree whose leaves are a sequence of turns. And a turn is again a tree, as is shown for the second turn. These rules for expanding the tree structures of transcripts are the document grammar. Writing a transcript, then, is like creating a structure by nesting trees, and then filling out the slots defined by the terminal leaves by entering data such as clock times or spoken prose.

Declaring the rules. SGML tags are defined through statements of the syntax governing their usage. To specify what a transcript is, you write a *document type definition* (DTD), which is a set of rewrite rules, written in SGML. These declarations appear at the beginning of a file, perhaps through a reference to another file that should be included at that point. The declarations specify how the file has been marked up. The rules defining the valid components transcript appear within a declaration like this:

```
<!DOCTYPE transcript [rules go here]>
```

A rewrite rule is called an *element* definition. The first element is always the document type itself, because it is the top node of the tree. So the first rule might say that a header followed by a body constitute a transcript. Subsequent rules would specify the components of the header

and body, and so on. The example below defines the components of a turn. The *generic identifier* is the name of the component and will appear in the tags (i.e., <turn>). The *minimization rule* says whether it is acceptable to omit either the start tag (<turn>) or end tag (</turn>), making the markup less verbose (see also Barnard, Hayter, Karababa, Logan, & McFadden, 1988b). If a turn can end only with the beginning of another turn or the end of an interview, then the end tag is redundant. The symbols -o mean that the start tag is required but the end tag is optional. Finally, the *content model*, within the parentheses, says that time, speaker, and speech elements are nested within turns; or that a turn is a (sub)tree structure like the one depicted in Figure 1.³

```
<!ELEMENT turn -o (speaker, time, speech)>
```

The elements in a turn must be defined in subsequent element declarations; for example,

```
<!ELEMENT speaker -- (#PCDATA)>
```

#PCDATA means "parsed character data"—that is, any characters *except* further SGML markup—so that the rule says that here we stop nesting structures and substitute some textual content. We could go farther and, for example, define sentences as elements of speeches. But if we do not need the computer to recognize sentences, we can leave them marked up with periods, spaces, and capitalization.

Advantages of Parsable Document Grammars

The DTD ensures that SGML-conformant tags will have a grammar known to the computer. Conformity to the guidelines is therefore testable, and by a procedure that is more efficient than sending the document to a human copy editor. The marked up transcript, including its DTD, can be submitted to an SGML parser, which can mechanically check whether the markup is syntactically correct. The ability to verify syntactic correctness makes it easy to implement a standard interchange format for social interaction transcripts.

The DTD provides a simple mechanism for distributing the rules for markup usage. The standard markup declarations would be placed in a file that could be circulated throughout a research community. To ensure that the rules were implemented, researchers would agree to include the public file when the document was parsed. This means that conformity to a common set of markup declarations would be mechanically checked for every transcript with this inclusion. A laboratory that would need to vary from the standard could do so only by (and simply by) adding declarations to the transcript that would override or extend the material in the public file. A transcript with modified declarations could still be processed by software that understood SGML, and the variations would be documented for future researchers.

Element Attributes

SGML tags can also contain a list of *attributes*—that is, data about the content of the marked textual segment.

The attribute data are encapsulated within the start and end delimiters of the tag. For example, we could define gender as an attribute of speakers, so that speaker tags would be written

```
<speaker gender=m>Patient</speaker>.
```

The attributes are created through declarations in the DTD, as in the following example, which shows an *element* speaker with a single *attribute*, gender. Additional attributes would be listed as additional rows enclosed by the guillemets. The declaration specifies the *acceptable values* of the attribute (either m or f) and the value that should be given as the *default*. In this case, `!!` IRED means that there is no default, and either m or f must be given by the transcriber.

```
<!ATTLIST speaker gender (m|f) #REQUIRED>
```

The element attribute facilities of SGML could transform the transcript into a powerful data structure. Attributes could be used to support (1) cross-references and hypertext, (2) the use of an expanded transcript as a common data structure for text and sequential codes, and (3) linkages across data channels in multimedia documents.

Cross-references and hypertext. In order to make a cross-reference from one transcript passage to another, we need a scheme for labeling locations within the text. To this end, researchers often number the turns within an interaction, or number the lines in a printed version. SGML allows one to declare attributes for elements that will uniquely identify each instance of the element. For example, one can declare an attribute that will allow one to number turns. This mechanism would facilitate the use of the computer in tracing and consulting cross-references within and between transcripts (Muhr, 1992).

For example, suppose an analyst wanted to add a note that clarified an interviewee's reference to a previous statement. Here is the original statement—the target speech of a cross-reference—where we see that speeches are now numbered with the id attribute `speechID`:

```
<speaker>Lois Lane</speaker>
<speech speechID=32 affect=sarcastic>Of course I know
that Clark Kent is a reporter!
</speech>
```

In the following example of speech with annotation and cross-reference, the speaker makes another assertion that reveals an interesting aspect of her background knowledge:

```
<speaker>Lois Lane</speaker>
<speechID=39 affect=shock>Superman couldn't be a
reporter!
<note>Subject seems unaware that Clark Kent is Super-
man. See above<xref speechID=32>.
</note>
</speech>
```

Assuming that `<note> . . . </note>` has been declared as an element of a speech, the transcriber can insert text annotating the event. The annotation contains a cross-reference from the note to the target text—that is, the pre-

vious speech—which provides evidence for the surmise. The advantage of identifying speeches and pointing to specific speeches with cross-references is that we could implement a hypertextual transcript (Conklin, 1987; TEI, 1990)—that is, a program that allows a reader to selectively view two or more cross-referenced views of a text. Suppose a person is reading a transcript rendered on a screen by a program that supports hypertext. She encounters the note in the second of the preceding examples and wishes to see the evidence that supports the inference that Lois Lane does not know that Clark Kent is Superman. She could instruct the computer to find the text identified in the cross-reference (i.e., the first example), retrieve it, and display it above the second example.

Coding text. The two preceding examples also contain attributes coding the subject's affect during the speech. This illustrates how codes pertaining to elements such as speeches or dialogue turns can be stored in a transcript. Sequential analysts (Bakeman & Gottman, 1986) would need these codes extracted from the transcript and placed in an external file for statistical analysis (for an example of a project based on aspirations to combine a micro-behavioral and a phenomenological perspective, see Stamp, 1991). Nevertheless, sequential analysts (Gottman & Roy, 1989) would benefit from keeping the codes stored within the transcript and extracting them when needed for purposes of analysis. To get the sequence of codes in an external file, all that would be required would be a program that scanned the text to locate each instance of a `<speech> . . . </speech>`, read the code, and wrote it on the external file. The advantage of using a transcript as the primary store for the sequential codes is that it retains a link between the sequential and textual data. This means that sequential analysts can easily recover the textual context surrounding a code.

The same effect could also be achieved with SGML by defining what the TEI (1990) calls a crystal—that is, a single tag that may contain information and that can be inserted into a text. For example, suppose that a code was placed in a margin, as in the following simulation of text processing:

```
Lois: Clark, you are the worst dresser in
Revelation    Metropolis. Let's look inside your closet.
OH! Clark! You are Superman?
```

Suppose that *revelation* is defined as a crystal with a single attribute, *mode*, with values of belief and disbelief. Then an SGML encoding would be as follows:

```
<speaker>Lois</speaker>
<speech>Clark, you are the worst dresser in Metropolis.
Let's look inside your closet. <revelation mode=disbelief>
OH! Clark! You are Superman?
</speech>
```

There are two advantages to the SGML approach. First, the code is placed directly into the text, where the event occurs. Second, the crystal structure allows additional information to be placed in the code, which could be a useful way to either implement a hierarchical coding scheme

or make a cross-reference within the transcript to a specific code.

SGML is most useful for textual structures that can be described as a tree, in which subordinate elements are cleanly nested within superordinate elements. As the affect coding example shows, it is easy to use SGML to represent codes that apply to a text segment that is a nested element within a tree structure. This means that one can readily create and add superordinate or subordinate codes to an existing document. For example, suppose that one wished to code <argument>s, where an <argument> is two or more contiguous <turn>s of conflict. To add markup for arguments, one could simply insert an <argument> prior to the <turn> marking the beginning of the first turn of the argument and insert </argument> following the </turn> of the last turn of the argument.

There are, however, some coding applications for which SGML will not be optimal. Suppose an analyst needs to identify textual segments that will not fit a tree structure. For example, one might wish to allow arguments to begin or end in the middle of a speaking turn. In this situation <turn>s do not nest within <argument>s; the codes are overlapping and nonhierarchical:

```
<turn>
<speaker>Lois</speaker>
<speech>Clark, it's so wonderful to see you <argument>
  Clark, would you please look at me when I'm talking
  to you!
</speech>
</turn>
<turn>
<speaker>Clark</speaker>
<speech>Lois, on Krypton I wouldn't have given you the
  time of </argument> Oops, I mean, I'm terribly sorry
  Lois, I was just thinking about the deadline on that story.
</speech>
</turn>
```

Beginning and ending tags for arguments can be defined following SGML syntax, but because they do not nest they cannot be parsed. Therefore, the validity of the use of these tags cannot be checked.

Multimedia linkages. Finally, the attributes for elements such as turns or speeches may eventually include references to locations in other media. There is considerable research in progress on the compression of digitized video images (Fox, 1989; Lippman & Butera, 1989). When this technology is mature, it will be possible to store large amounts of video data on media such as optical disks that would allow fast retrieval of video segments. This would allow us to specify speaking turn attributes that identified the starting and ending video frames corresponding to the start and end of the turn. This would allow us to play the video segment and simultaneously display the text, sequential codes, and anything else pertinent to that coded element.

Example: Coping With Temporal Complexity

As an example of an application of SGML to a more complex transcription task, consider the problem of rep-

resenting how speaking "turns" actually overlap and jam each other in real time. SGML imposes a formal structure on discourse, but spoken prose is gloriously refractory to such formality. For example, an important difference between spoken and transcribed dialogue is that whereas the two participants will frequently produce overlapping or parallel speeches, most transcripts are formatted as single streams of text, composed of discrete alternating turns. SGML-based conventions would be unsuitable for marking up transcripts if they did not allow us to mark the violations of temporal structure characteristic of interviews. A good test of this is to examine some of the markup already developed for complex turn taking, and to see whether it can be expressed in SGML. This poses problems for any typographical representation of dialogue, whether computerized or simply typewritten. The canonical typographical representation of dialogue is a single sequence of alternating speaking turns. Spoken dialogue, however, is full of ellipses, interjections, and interruptions that are difficult to represent, because standard written prose lacks the resources to express these temporal relations. Researchers have resorted to various strategies for transcribing text, such as inventing new notation (Sacks, Schegloff, & Jefferson, 1974), formatting the dialogue in parallel columns to allow overlapping speaking turns (Rogoff & Gardner, 1984), or simply filtering the incongruous temporal details out of the transcript.

One ambitious scheme for marking up transcripts to capture such complications was devised by Gail Jefferson (Sacks et al., 1974). To illustrate how a workable human system may not be a workable computer system, I will discuss how a computer might try to cope with Jefferson's ingenious system.⁴

For example, consider the following notation for overlapping speech (adapted from Psathas, 1979).⁵ The // marks the point in V's speech where M's speech begins:

V: Th'guy says tuh me- 'hh my son // didid.
M: Wuhjeh do:.

Now suppose that M interjects a comment twice:

V: I // left my garbage pail in iz // hallway.
M: Vi:c,
M: Victuh.

The convention here is that M's first statement maps to the first //, and his/her second, to the second //. This is likely to pose a problem for a computer, because Jefferson has borrowed the markup she ordinarily uses for turns, but "Vi:c, Victuh" is not two speaking turns, as the comma indicates. This too could be mastered by a computer, but consider the subtlety of interpretation required. The first M: in the second example not only marks the beginning of an overlapping speaking turn, but also identifies the target of a cross-referential link anchored at the first //, while the second M: serves only as a cross-reference and does not mark a turn. One could get a computer to use the context to correctly interpret the different meanings of the two M:s, perhaps by maintaining a stack of // occurrences. If the stack has more than one

item, the computer would expect more than one M:, of which only the first marks a new turn. So far so good, but what happens when a transcriber inadvertently introduces an extra / / into the text? Answer: the entire markup system for turn taking breaks down.

This example shows that presentational markup for complex discourse is unlikely to survive the computerization process. Although word processors can reproduce Jefferson's markup, it seems unpromising to construct a computer program that could interpret it. Therefore, this markup system would likely be invisible to a computer, down to the organization into turns. The example also suggests that SGML provides sufficient discipline and clarity to construct a markup system that can, on the one hand, manage the temporal complexities of interaction and, on the other hand, do so without usurping any of the ordinary presentational markup of punctuation, underlining, and the like.

Several strategies could be adopted to handle temporal complexity. For example, we could define an element that marks the location in one person's speech where an interjection from another has occurred. For example, let `<interrupt>` (with attribute `interruptID`) mark the point in a person's speech where another speaker enters. We could write the element specification for the turn to allow turns to contain several speeches, and include an attribute in a speech that identified that speech as an interruption. A cross-reference could be defined that pointed to the corresponding `<interrupt>`. The temporal sequence of events could then be resolved unambiguously.

Recommendations

Interaction researchers using transcripts should consider the development of a standard encoding format for machine-readable text. Specifically, they should consider joining the TEI and developing a set of markup conventions that is consistent with the *Guidelines* (TEI, 1990), extended to include text objects important to social scientists concerned with social interaction. The development of such public DTDs is exactly the point of the TEI, and their work and model will ease the efforts of any group with similar goals. The experience of the TEI suggests, however, that one ultimately encounters limits in the degree to which the structural complexity of human conversation can be modeled in a formal language such as SGML. Therefore, it is by no means certain that every coding problem will have a valid SGML solution. The utility of this approach can only be established through empirical trial.

DISCUSSION

Public DTDs would make it feasible for communities of researchers to design and collectively implement a common transcript format. Because conformity to the standard can be mechanically checked and variations from the standard must be explicitly declared, the standard would be more than a wish list. SGML markup enables the com-

puter to unambiguously identify transcript components, and the parsable grammar guarantees that the components can be located at a known place in a logical tree. This means that transcripts will be in a format that provides a foundation for writing database programs that can retrieve and manipulate variable records. With such programs, researchers could efficiently create new transcripts combining excerpts gathered from deep and wide text databases.

It is clear, however, that there are also substantial costs in using descriptive markup and in using SGML in particular. It may require more effort to create a transcript that is marked up in SGML. The Department of Defense and the American Association of Publishers, however, have both adopted SGML as a standard, which has created a significant and rapidly evolving market for SGML software tools. These tools include applications for creating documents with SGML markup that purportedly save much of the additional keyboarding required to represent "Round up all the usual suspects." as `<speech> Round up all the usual suspects </speech>`. Even so, SGML literacy would become yet another technical skill that researchers would need to master.

Progress toward a better format for the encoding of transcribed interactions is an urgent task. Many of the current systems for the computerization of social interactions reflect the technology of the microcomputer revolution of the early 1980s. Microcomputers allowed individual scientists and small research projects to acquire computers and, because they were ideally suited for word processing, employ them to process their texts efficiently. The software resulting from these projects generated incompatible databases, but this was not perceived to be a problem. As scientific communication shifts from print to computerized media, it will be essential that data be encoded in, or translatable to, the emerging electronic publication and network communication standards (Gardner, 1990, 1992).

REFERENCES

- BAKEMAN, R., & GOTTMAN, J. M. (1986). *Observing interaction: An introduction to sequential analysis*. New York: Cambridge University Press.
- BAKEMAN, R., & QUERA, V. (1992). SDIS: A sequential data interchange standard. *Behavior Research Methods, Instruments, & Computers*, **24**, 554-559.
- BARNARD, D., FRASER, C., & LOGAN, G. (1988a). Generalized markup for literary texts. *Literary & Linguistic Computing*, **3**, 26-31.
- BARNARD, D., HAYTER, R., KARABABA, M., LOGAN, G., & MCFADDEN, J. (1988b). SGML-based markup for literary texts: Two problems and some solutions. *Computers & the Humanities*, **22**, 265-276.
- BRYAN, M. (1988). *An author's guide to the Standard Generalized Markup Language*. Wokingham, U.K.: Addison-Wesley.
- CONKLIN, J. (1987). Hypertext: An introduction and survey. *IEEE Computer*, **20**, 17-41.
- CONRAD, P., & REINHARZ, S. (1984). Computers and qualitative data. *Qualitative Sociology*, **7**, 3-15.
- COOMBS, J., RENEAR, A., & DEROSE, S. (1987). Markup systems and the future of scholarly text processing. *Communications of the ACM*, **30**, 933-947.
- ERICSSON, K. A., & SIMON, H. A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

- FOX, E. (1989). The coming revolution in interactive digital video. *Communications of the ACM*, **31**, 794-801.
- GARDNER, W. (1990). The electronic archive: Scientific publishing for the '90s. *Psychological Science*, **1**, 333-341.
- GARDNER, W. (1992). Prospects for the electronic publication of scientific journals. *Current Directions in Psychological Science*, **1**, 75-78.
- GOTTMAN, J. M., & ROY, A. (1989). *Sequential analysis: A guide for behavioral research*. New York: Cambridge University Press.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (1986). *ISO 8879-1986 (E). Information processing—Text and office systems—Standard Generalized Markup Language (SGML)* (1st ed.). Geneva: International Organization for Standardization.
- JAMES, J. M., & SANDERSON, P. M. (1991). Heuristic and statistical support for protocol analysis with SHAPA Version 2.01. *Behavior Research Methods, Instruments, & Computers*, **23**, 449-460.
- LAMPORT, L. (1986). *LaTeX: A document preparation system*. Reading, MA: Addison-Wesley.
- LEVENSON, R. W., & GOTTMAN, J. M. (1983). Marital interaction: Physiological linkage and affective exchange. *Journal of Personality & Social Psychology*, **45**, 587-597.
- LEVENSON, R. W., & GOTTMAN, J. M. (1985). Physiological and affective predictors of change in relationship satisfaction. *Journal of Personality & Social Psychology*, **49**, 85-94.
- LIPPMAN, A., & BUTERA, W. (1989). Coding image sequences for interactive retrieval. *Communications of the ACM*, **32**, 852-860.
- MACKAY, W., & DAVENPORT, G. (1989). Virtual video editing in interactive multimedia applications. *Communications of the ACM*, **32**, 802-810.
- MISHLER, E. (1986). *Research interviewing: Narrative and context*. Cambridge, MA: Harvard University Press.
- MUHR, T. (1992). ATLAS/ti—A prototype for the support of text interpretation. *Qualitative Sociology*, **14**, 349-371.
- PSATHAS, G. (Ed.) (1979). *Everyday language: Studies in ethnomethodology*. New York: Irvington Publishers.
- RENEAR, A. H., DURAND, D. G., & MYLONAS, E. (1992). Overlapping hierarchies of text objects: Refining our notion of what text really is. In *Association for Literary and Linguistic Computing/Association for Computing in the Humanities* (pp. 200-203).
- ROGOFF, B., & GARDNER, W. (1984). Adult guidance of cognitive development. In B. Rogoff & J. Lave (Eds.), *Everyday cognition: Its development in social context* (pp. 95-116). Cambridge, MA: Harvard University Press.
- SACKS, H., SCHEGLOFF, E., & JEFFERSON, G. (1974). A simplest systematics for turn-taking in conversation. *Language*, **50**, 731-734.
- SCHEGLOFF, E. A. (1987). Analyzing single episodes of interaction: An exercise in conversation analysis. *Social Psychology Quarterly*, **50**, 101-114.
- STAMP, G. H. (1991). Family conversation: Description and interpretation. *Family Process*, **30**, 251-263.
- TESCH, R. (1990). *Qualitative research: Analysis types and software tools*. New York: Falmer Press.
- TESCH, R. (Ed.) (1991a). Computers and qualitative data II. Special issue, Part 1. *Qualitative Sociology*, **14**, 225-285.
- TESCH, R. (Ed.) (1991b). Computers and qualitative data II. Special issue, Part 2. *Qualitative Sociology*, **14**, 289-384.
- TEXT ENCODING INITIATIVE (1990). *Guidelines for the encoding and interchange of machine-readable texts* (Draft: Version 1.0).

NOTES

1. I thank a reviewer for suggesting this point.
2. In 1992, for example, to translate a file from the WordPerfect Macintosh (2.0) version to the UNIX (5.0) version changes the layout of the document.
3. One more very useful feature of SGML deserves mention: its ability to support more than one concurrent markup hierarchy. A simple example of concurrent hierarchies would be two different segmentations of an interview, one based on speaking turns and the other based on the non-verbal behavior of the partners. Because these segmentations need not overlap, they could not be nested within each other. The concurrency of SGML would allow one to define two alternative document type hierarchies, and to select which one to use when the document was processed. Whether this feature is sufficiently powerful to represent the complexity of real text has been disputed (Renear, Durand, & Mylonas, 1992).
4. No criticism of Jefferson is intended; she designed her notation for human readers, not computers.
5. Some of Jefferson's notation for indicating stress is omitted.

(Manuscript received June 1, 1992;
revision accepted for publication December 16, 1992.)