As-Living-Thing that includes 12 extended metaphors. However, no taxonomies are given for the other very general core metaphors used in MIDAS, which are **Location-Metaphor, At-State, Have-State, Container-Metaphor, Kill-Metaphor**, and **Eating-Metaphor**. Moreover, there is little discussion of the relationship between these core metaphors.

The third question is: is there some way to reduce the enormous number of metaphorical interpretations that MIDAS seeks? Step 3 of the metaphor interpretation algorithm given on page 95 states that MIDAS collects "all possible interpretations, both metaphorical and literal," including presumably direct application of the metaphors in MIDAS's knowledge base plus the use of MIDAS's metaphor extension techniques. Metaphors are sought where there are no constraint violations (p. 104). This is a vast amount of processing, and remember that MIDAS only uses 70 or so metaphors — a larger system might contain hundreds. Martin might reply that realistic metaphor interpretation does involve an enormous amount of processing. He may be right.

References

Barnden, John A. (1989). "Belief, metaphorically speaking." In *Proceedings, 1st International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann.

Barnden, John A. (1990). "Naive metaphysics: A metaphor-based approach to propositional attitude representation (unabridged version)." Memorandum MCCS-90-174, Computing Research Laboratory, New Mexico State University.

Carbonell, Jaime G. (1982). "Metaphor: An inescapable phenomenon in natural language comprehension." In *Strategies for natural language processing*, edited by Wendy G. Lehnert and Martin H. Ringle, 415–434. Lawrence Erlbaum Associates.

Jacobs, Paul S. (1987). "Knowledge-intensive natural language generation." *Artificial Intelligence*, **33**, 325–378.

Lakoff, George and Johnson, Mark (1980). *Metaphors We Live By*. The University of Chicago Press.

Norvig, Peter (1989). "Marker passing as a weak method for text inferencing." *Cognitive Science*, **13**, 569–620.

Dan Fass is a visiting fellow at the Centre for Systems Science, Simon Fraser University, Canada. Before this, he worked for three years at the Computing Research Laboratory, New Mexico State University, U.S.A. His Ph.D. is from the University of Essex, England. His research interests include the computer understanding of metaphor and metonymy, lexical ambiguity, machine-readable dictionaries, ill-formed input, and beliefs. Fass's address is Centre for Systems Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6; e-mail: fass@cs.sfu.ca.

Practical SGML

Eric van Herwijnen
(CERN)

Dordrecht: Kluwer Academic Publishers, 1990, xviii + 307 pp.
Paperbound, ISBN 0-7923-0635-X,
$39.95

Reviewed by
Carol Van Ess-Dykema
U.S. Department of Defense and Carnegie Mellon University

1. Introduction

This review begins with background information on what SGML is. This is followed by a brief synopsis of SGML's benefits and extensibility as an information management tool for members of the computational literary and linguistic communities. The review then describes the current status of the Text Encoding Initiative's SGML draft *Guidelines* for computational literary and linguistic use. It identifies two projects within the computational linguistic community that have made commitments to encode their corpora and the results of their research with SGML. The final part of the review evaluates *Practical SGML*, focusing on its effectiveness as a reference manual for computational literary and linguistic practitioners who wish to learn to use SGML.

2. What Is SGML?

The Standard Generalized Markup Language, commonly referred to as SGML, is an advanced tool in the science of information management that provides its users data portability through standardization. It is the International Standards Organization (ISO) standard for document description (ISO 8879). The ISO standard defines SGML as "a language for document representation that formalizes mark-up and frees it of system and processing dependencies." It is an abstract language developed to facilitate text interchange in the form of computer-readable data, intended primarily for use by publishers but with application for the computational literary and linguistic communities as well. SGML provides *standardized* mark-up conventions so that interpretation for structural or typographic purposes need not be improvised for each document every time it is transmitted (Fought and Van Ess-Dykema 1990).

SGML-encoded documents are transportable from one hardware or software environment to another without loss of information. An SGML document can be transported from your computer to a different computer via a network, diskette, or tape. SGML-encoded data can be transferred from a text formatter to a database. Van Herwijnen states in the preface to *Practical SGML* that SGML allows for the interchange of large amounts of complex data and for easy access to them. With SGML, data become independent of their medium. Storage is no longer restricted to paper, but can be in different forms such as in a database or on optical disk (p. ix).

SGML is not a text formatter like TEX that many computational linguists are familiar with. TEX is a computer program that performs tasks of page makeup and typesetting that were traditionally carried out manually. Text formatting languages like TEX use special characters such as \, {, and & that may be translated incorrectly when a document passes from one computer to another (pp. 3, 16–17). SGML mark-up, in contrast, uses system-independent tags to indicate the structural composition of a document. For example, SGML mark-up for a memorandum includes, but is not limited to, the following: ⟨Memo⟩, ⟨To⟩, ⟨From⟩, ⟨Body⟩, ⟨P⟩(paragraph), ⟨Q⟩(quotation) (p. 33).

3. Extensibility to Literary and Linguistic Computing

The present diversity of encoding schemes used for literary and linguistic texts makes it difficult to move texts from one software program to another. Researchers who receive texts from others must decipher the texts and convert them into their local encoding scheme before they can use them. As machine-readable texts and encoding schemes proliferate, the need for a common scheme becomes more pressing (Text Encoding Initiative 1988).

SGML mark-up conventions designed for literary and linguistic use have been drafted by members of the Text Encoding Initiative (described below). This discipline-specific SGML mark-up will allow for the interchange of existing literary and linguistic data and for the encoding of new documents. Computational literary and linguistic practitioners will, moreover, be able to encode the results of their research on texts with SGML mark-up; for example, encoding the metrical structure of verse and the analysis of sentence syntax.

SGML mark-up conventions developed for the literary and linguistic communities' local processing needs will be able to be used by programs that do the following:

- edit texts (e.g., word processors, syntax-directed editors, hypertext systems),
- format and print texts (word processors, batch-oriented formatting programs like Scribe, Script, Runoff, roff, or TEX),
- load texts into free-text retrieval databases or conventional databases,
- unload texts from databases as search results or for export to other software,
- search texts for words or phrases,
- perform content analysis on texts,
- collate texts for critical editions,
- scan texts for automatic indexing or similar purposes,
- parse texts linguistically,
- analyze texts stylistically,
- scan verse texts metrically,
- link words of a text to images of the objects named by the words (as in a hypertext English or foreign-language teaching system) (Sperberg-McQueen and Burnard 1990).

4. Text Encoding Initiative

The Text Encoding Initiative (TEI) is a cooperative undertaking of the Association for Computers and the Humanities (ACH), the Association for Computational Linguistics (ACL), and the Association for Literary and Linguistic Computing (ALLC). The Initiative has already formulated and disseminated the first draft *Guidelines for the Encoding and Interchange of Machine-Readable Texts*, intended for literary, linguistic, historical, or other textual research. The format is as far as possible hardware- and software-independent, rigorous in its definition of textual objects, easy to use, and compatible with existing standards (Text Encoding Initiative 1988). Four working committees addressed the questions of text documentation, text representation, text analysis and interpretation, and metalanguage issues, in the preparation of the draft guidelines.

The first version of the *Guidelines*, published in 1990, will be revised and extended over the next two years. The final version will be published in 1992.[1] The draft guidelines are incomplete and imperfect. They need to be evaluated by a large portion of the computational literary and linguistic communities. The final version will most likely still leave some individuals grumbling, but the communities at large are certain to gain.

The TEI does not expect that all members of the literary and linguistic communities will immediately encode their collections of texts with SGML. The benefits of hardware- and software-independent interchange of data without loss of information, however, is expected to produce many enthusiastic SGML users in industry, governments, universities, and research centers.

5. Current Linguistic Projects Using SGML

There are several projects within the computational linguistics community that have made commitments to encode their corpora and the results of their research analyses with SGML.

The material in the ACL Data Collection Initiative text corpus will be encoded in a standard form based on SGML. Over time, the Initiative members hope to be able to incorporate annotations reflecting consensually approved linguistic features, such as part of speech and various aspects of syntactic and, perhaps, semantic structure. The text corpus is housed with Mark Liberman in the Department of Linguistics at the University of Pennsylvania. Both the encoding and the annotations will be coordinated with the work of the Text Encoding Initiative.

The goal of the "Tree Bank of Written and Spoken American English" project, also at the University of Pennsylvania, is to annotate millions of sentences with part-of-speech assignment, skeletal syntactic parsings, intonational boundaries for spoken language, and other forms of linguistic information that can be encoded consistently and quickly. The project will be coordinated with the work of the TEI's Committee on Text Analysis and Interpretation, which is concentrating initially on developing SGML mark-up for linguistically salient features of texts. Mitch Marcus in the Department of Computer and Information Science is housing the data of the project (Association for Computational Linguistics, Data Collection Initiative 1989a, 1989b).

6. How to Learn to Use SGML

Practical SGML contains 13 chapters, organized into three parts. Van Herwijnen states in the preface that Part I, "Getting Started with SGML," is for authors, document managers, programmers, and everyone who needs an introduction to SGML. Part II, "Advanced SGML," is for document managers, programmers, and interested authors; Part III, "SGML Implementations," is for application programmers.

---

1 Interim drafts of the *Guidelines* are available to interested members of the computational literary and linguistic communities for review and comment, upon request from either of the following addresses:

- Lou Burnard, Oxford University Computing Service, 13 Banbury Rd, Oxford OX2 6NN, UK. Fax: +44 (865) 273275; e-mail: lou@vax.oxford.ac.uk.
- C.M. Sperberg-McQueen, Computer Center MC 135, University of Illinois at Chicago, Box 6998, Chicago, IL 60680, U.S.A. Fax: (312) 996-6834; e-mail: U35395@uicvm.bitnet or U35395@uicvm.cc.uic.edu.

There is no charge for the first copy sent to any one address.

Van Herwijnen says that after reading Part I (Chapters 1–5), the reader should know, among other things:

- something about the history of SGML,
- the basic ideas of SGML,
- what its advantages are,
- how to mark up a document,
- how to read and write a Document Type Definition (DTD),
- what the functions of a parser are,
- how to manage SGML.

Part II (Chapters 6–8) presents:

- some formal aspects of the SGML language,
- how data characters are distinguished from mark-up,
- what the reference concrete syntax is,
- what the SGML declaration is,
- how to create tagged documents using various editors or tagging systems.

Part III (Chapters 9–13) explains:

- what software components exist in an SGML environment,
- how to create SGML documents with non-SGML word processors,
- why SGML editors are important,
- some examples of parsers,
- how to translate SGML tags into application procedures,
- what Computer-Aided Acquisition and Logistics Support (CALS) is,
- how to add SGML documents to a database,
- how you can use SGML to describe Electronic Data Interchange (EDI).

Each of the book's 13 chapters includes a bibliography. Selected chapters contain comprehension questions as well.

Appendix A provides the answers to the comprehension questions. In Appendix B, van Herwijnen describes all aspects of the electronic publication of *Practical SGML* itself. He includes a description of how he moved the manuscript in SGML form over a computer network between author and publisher, the former with an EBCDIC system and the latter with an ASCII system, until the final version was produced. Appendix C presents the Document Type Definition that van Herwijnen wrote for the book. Appendix D contains entity definitions for use with the text formatter TEX. Appendix E explains the International Standards Organization SGML standard, ISO

8879. The book contains a glossary of terms and definitions presented in the text, and an index.

An SGML user needs to know what is contained in the three parts of an SGML document: the SGML declaration, the document type definition, and the document instance. Chapters 2, 3, 4, and 6 of *Practical SGML* explain these three parts admirably. No successful document interchange or application processing can occur without these three parts included in or along with the document. As these chapters explain, the *document instance* is the part of the SGML document that contains the marked-up textual data. It needs to be translated into processing commands before it can be printed. The *document type definition* (DTD) defines the mark-up rules for a given class of documents. Programs called SGML *parsers* analyze and check that the mark-up in a specific document satisfies the rules defined by the DTD. This means that different documents of the same type can be processed in a uniform way. The SGML *declaration* defines which characters are used in a document instance, which syntax the DTD is written in, and which SGML features are used.

An SGML environment comprises three subsystems. New SGML users may already have available one or more of them in their local computing environment. The three are: an input system for developing SGML documents, a parser for checking SGML documents, and some system to process (translate) the parsed SGML documents. Van Herwijnen discusses these subsystems in Chapters 9 and 10. He correctly emphasizes that the parser should verify every document before a program translates it for processing or before it is exported to a different computer. Since SGML documents are processor-independent, they need to be translated into specific commands for any given formatter. The translation should be done automatically by a program.

An SGML user's source documents may originate from optical character-recognition scanners, ASCII files, printer files, and word processor output. Commercial SGML editors are the ideal SGML *input system*. However, for those literary and linguistic practitioners who are interested in learning SGML but who find themselves without such an input system at their disposal, van Herwijnen states that the following editors and word processing systems can also be used: a simple text editor, an editor that formats, or a combination word processor and stylesheet. He explains that in the case of a simple text editor, the SGML mark-up tags are added by hand in the editor, or later by relying on the parser to understand typewriter conventions defined in the DTD. In this situation the SGML parser adds the mark-up while it parses the document. An easier way to insert the SGML mark-up is to use an editor that is able to format text on the screen. Typographical styles (e.g., font) for all document elements defined in the DTD can also be made and grouped together in macros. Using the latter, a structure may be enforced on any document created with the word processor. Mark-up may be added by a program afterwards, or by the parser if the DTD contains typewriter conventions (p. 182).

The book, unfortunately, contains many copyediting infelicities. One wishes there had been as much care taken in editing the text as there obviously was in developing the material contained in the text and in inserting the SGML mark-up into the text. None of the infelicities precludes the book's effectiveness.

I recommend *Practical SGML* to computational literary and linguistic practitioners who may wish to learn SGML. It should serve as an excellent reference manual for those desiring the benefits of this new tool in information management. Van Herwijnen is the leader of the text processing section at CERN, the European Laboratory for Particle Physics, Geneva, where there are a large number of end-users of varied experience. I agree with the writers of the book's foreword, who state that it is obvious

that van Herwijnen has written the book on the basis of his own first-hand experience with SGML (p. vii).

I suggest that interested practitioners use *Practical SGML*, together with the computational literary and linguistic SGML draft *Guidelines* developed by the four committees of the Text Encoding Initiative, in their preparation of SGML documents. Consult *Practical SGML* first, and then the *Guidelines*. A third source one may wish to consult is *SGML: An Author's Guide to the Standard Generalized Markup Language* by Martin Bryan (1988). Then roll up your sleeves and dig in. Insert the SGML tags and attributes into your document. Create your SGML declaration. Write your DTD. And don't forget to have the parser validate your mark-up before interchanging the document with another computer or translating it for processing.

References

Association for Computational Linguistics, Data Collection Initiative (1989a). *The Finite String*, **15**(1), March 1989, 1–2.

Association for Computational Linguistics, Data Collection Initiative (1989b). *The Finite String*, **15**(4), December 1989, 46–47.

Bryan, Martin (1988). *SGML: An Author's Guide to the Standard Generalized Markup Language*. Addison-Wesley.

Fought, John and Van Ess-Dykema, Carol (1990). "Toward an SGML document type definition for bilingual dictionaries." Technical Report TEI AIW20, Text Encoding Initiative.

International Standards Organization (1986). "Information processing — Text and office systems — Standard Generalized Markup Language (SGML) (ISO 8879)." ISO.

Sperberg-McQueen, C. M. and Burnard, L., eds. (1990). *Guidelines for the Encoding and Interchange of Machine-Readable Texts*. Draft Version 1.0. Text Encoding Initiative.

Text Encoding Initiative (1988). "Text encoding initiative: Initiative for text encoding guidelines and a common interchange format for literary and linguistic data." Document TEI J3. Text Encoding Initiative.

Carol Van Ess-Dykema is a member of the Dictionary Encoding Subcommittee of the Text Representation Committee of the Text Encoding Initiative. She is a computational linguist at the U.S. Department of Defense. She presently holds a postdoctoral fellowship at the Center for Machine Translation, Carnegie Mellon University. Van Ess-Dykema's address is Center for Machine Translation, Smith Hall, Room 109, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213; e-mail: vaness@nl.cs.cmu.edu.