On Lean Messaging with Unfolding and Unwrapping for Electronic Commerce
Author(s): Steven O. Kimbrough and Yao-Hua Tan
Reviewed work(s):
Electronic Commerce (Fall, 2000), pp. 83-108
Published by: M.E. Sharpe, Inc.
Stable URL: http://www.jstor.org/stable/27750960
Accessed: 01/03/2013 10:32

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at
http://www.jstor.org/page/info/about/policies/terms.jsp

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of
content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms
of scholarship. For more information about JSTOR, please contact support@jstor.org.

M.E. Sharpe, Inc. is collaborating with JSTOR to digitize, preserve and extend access to International Journal
of Electronic Commerce.
On Lean Messaging with Unfolding and Unwrapping for Electronic Commerce

Steven O. Kimbrough and Yao-Hua Tan

ABSTRACT: EDI (electronic data interchange) messages are notoriously lean and difficult (or impossible) to interpret without additional information. The authors acknowledge the many criticisms of the EDI protocols, but argue that there is something basically correct, even inevitable, in the leanness of EDI messages. They present a framework that describes how EDI messages are interpreted, and indeed must be interpreted. “Unwrapping” and “unfolding” of messages are the central elements. These concepts are discussed in detail, and the article demonstrates how to exploit them in formalizations for electronic commerce. In particular, it shows how Kimbrough’s lean-event semantics for speech acts, and Tan and Thoen’s theory of directed obligation can be fit naturally and fruitfully into this framework, and to each other. Much remains to be done, but the progress in formalization in evidence here should be generalizable.

KEY WORDS AND PHRASES: Business messaging, deontic logic, electronic commerce, electronic data interchange (EDI), formal language for business communication (FLBC).

Context

Most observers of electronic data exchange (EDI), and indeed most participants, would agree that present technology and standards for computer-to-computer exchange of structured information are importantly, even fundamentally, deficient.1 Among other things, costs are too high, the daunting first-trade problem discourages small and medium-sized enterprises (SMEs) from participating, too much manual intervention is needed even for well-established systems, established protocols are not suited to the Web, there is no semantics or existing protocol (or only incomplete and confused ones), and so on (cf. [2, 15, 19, 33]). All this and more is part of the received wisdom today, and the authors largely agree.

What to do? The usual move in information systems, when faced with a puzzling application problem, is to restructure the application domain for more effective computerization. A formal theory, including a formal semantics, would seem to be in order. Why?

1. Computerization requires that the application domain be structured and formalized. This may occur explicitly, in the form of a theory, or implicitly and only in the minds of the programmers. Clearly, the explicit route is preferable.

2. Explicit structuring and formalization allow users to know, in an important sense, what they are doing. A formal semantics for a domain provides insight and understanding that can be used to revise and apply theories, thereby making it possible to specify and test requirements independently of a particular implementation.
For example, whether or not there is an implementation, a logic model of a purchase order can be examined and tested for the inferences it supports. This provides ways to validate specific implementations.

3. Formalization and structuring also facilitate the extension of representations and the designing of rules for generating and processing messages. This will be illustrated in the sequel. For now, simply note that structuring and formalization ought to help with interpreting EDI messages but also with the rules and policies that do the interpreting.

Again, these are widely accepted views with which the authors agree. But how to proceed from here? Different researchers have different approaches to structuring and formalizing the EDI domain. Some favor a logic-based approach (e.g., Kimbrough, Moore, and Covington (see [3, 4, 5, 14, 15, 16]). Others prefer an approach based on XML (e.g., [20, 32, 33]), Petri nets (Lee and his group at EURIDIS; see [1, 17]), or some other formalism. Even given a formal strategy, there will, no doubt, be different substantive approaches. Besides the call for formalization and rigor, are there any other principles that can be invoked to guide these investigations? The answer is yes, and the main purpose of this paper is to present and argue for a set of principles of this kind that the authors call lean messaging with wrapping, unfolding, and unwrapping. The authors believe that existing EDI systems already embody these principles. This is so for good reasons, and thus, even with improved messaging standards, the principles should remain in force. In fact, a main reason for introducing improvements is to avail EDI systems of the benefits of these principles.

Lean Messaging with Wrapping

The Basic Schema

Consider in the abstract how business messaging—and EDI messaging in particular—is effected. The basic EDI messaging schema is illustrated in Figure 1. When two organizations, s and r, wish to do business with EDI, they typically begin by negotiating an interchange agreement. This is a contract that, among other things, specifies the rules of trade between s and r, including which EDI protocols and transaction sets are to be used, and how EDI messages are to be interpreted. Once this contract is in place, the sender of a message can draw upon its own knowledge bases, its background knowledge (e.g., relevant laws and administrative rules), and the interchange agreement to formulate a specific message, μ (utterance). Similarly, the receiver of a message draws upon its own knowledge bases and background knowledge (including the interchange agreement) in order to interpret and act upon the incoming message, μ.

A simple example—requesting and responding to a price quote—will focus the discussion, and (without loss of generality) bear the points just made.

Figure 2 presents a simple, pre-EDI, paper-oriented quotation form. This
form can be, and is, used for routine business-to-business transactions. It is the sort of thing that EDI protocols represent, with the aim of replacing paper forms with computer-to-computer automated messaging. Actual EDI protocols, fully specified (e.g., in X12 and EDIFACT), are considerably more complex and fraught with options than might be thought after an examination of Figure 2.

Figure 3 shows a valid request-for-quotation EDI message in the X12 standards. Figure 4 shows its approximate translation in plain English. Finally, Figure 5 shows the inevitable purchase-order form. Together, Figures 1-5 will serve as the basic reference examples for the discussion that follows. Although they have been simplified for the sake of the discussion, there is no loss of generality with regard to the issues at hand.

Here now are some reflections on this basic schema.

**Leanness**

By virtue of the messaging schema, a certain leanness attends EDI messaging, and, indeed, business messaging in general. In fact, one of the characteristic features of business messages (including EDI) is their unintelligibility without substantial reference to context. EDI messages present data structures that are basically stripped down, with little or nothing in them explaining the relations among the data elements constituting the message. For example, in the Request for Quotation EDI message (see Figure 3), only a code in the header indicates that it is a data structure for representing an RFQ. This is the case generally in both the X12 and EDIFACT standards, as well as the SWIFT protocols. It is also true of standard paper forms for doing business (see Figures 2 and 5). The types of information given in these message data structures are the names and addresses of buyer, seller, and involved banks for payment of the goods, payment method, delivery terms, and so forth. But neither in the EDI messages nor in the standard business forms is there an explanation of how these different roles relate to one another. For example, the information that the seller is the one who delivers goods to the buyer, and that the buyer is the one who pays for the goods, is not (explicitly) represented in the message data structures.

There are several reasons for the minimalist representation of information
TO: | QUOTE DATE
---|---
| YOUR REQUEST NO.
| OUR QUOTATION NO.

SHIPPED TO: | THIS QUOTATION VALID ONLY FROM TO
---|---

DELIVERY DATE PROMISED | DELIVERY VIA | FOB | TERMS
---|---|---|---

<table>
<thead>
<tr>
<th>ITEM</th>
<th>QUANTITY ORDERED</th>
<th>DESCRIPTION</th>
<th>UNIT COUNT</th>
<th>UNIT PRICE</th>
<th>TOTAL AMOUNT</th>
</tr>
</thead>
</table>

SPECIAL NOTES: | DATE: | APPROVAL SIGNATURE
---|---|---

THANK YOU FOR REQUESTING A QUOTATION FROM US. WE ARE PLEASED TO PROVIDE YOU WITH THIS INFORMATION. WE THANK YOU FOR YOUR PAST BUSINESS, AND WE LOOK FORWARD TO SERVING YOU IN THE FUTURE.

**Figure 2. Generic Quotation Form**

After http://entrepreneurmag.com/formnet/

in EDI (and other business-to-business) messages. One is, of course, efficiency of data transfer. The fewer bits an EDI message contains, the less need it has for resources. A second reason is that EDI messages are internationally standardized. Providing only the bare data and omitting textual explanations of the relations among the data reduces the problem of translation from one language, or legal system, to another. And there is a governing (heuristic) principle here: Only send news. What is reasonably constant is treated as background, as context, and belongs to the wrapper. Messages are for new information.

EDI (and business-to-business messaging in general) is about much more than the electronic transfer of messages. Computer-mediated, automated processing of messages, for sending or upon receipt, is at the heart of the matter.
Without automated creation of, and response to, electronic messages, there would be little point to EDI. One might as well use plain e-mail.

When EDI messages operate leanly, however, and omit so much information, understanding and processing them becomes a nontrivial exercise. Humans reading paper forms (see Figures 2 and 5) have background knowledge and training that allows them to interpret a message, and know, for example, that the sender of a purchase order (presumably) expects to receive certain goods and to pay for them in a certain way. A trained person could respond to a request for a quotation by ascertaining that the sender should be responded to, checking with the proper source in-house, and forwarding a reply to the sender (say by filling out a form like the one in Figure 2).

Automated processing of an EDI message requires that the computer system be able to store in its own database such important information as the name of the sender, the ID of the message, what the sender wants, and so on. Automated processing also requires that the computer system be able to initiate further action. A very simple example would be a computer system that responds to a purchase order by sending information relevant for the production of the requested goods to the production department, advance notification of the impending shipment of the goods to the carrier, and an invoice, after the goods are shipped, to the buyer. Depending on the delivery terms, the computer might also arrange insurance of the goods while in transit.

All of this, whether done by people or by computer systems, requires (among other things) that:

[1] [ST*840*159
[2] [BQT*00*Q47391*820430
[3] [N1*SE*X, Inc.
[4] [N1*BY*Y Co.
[5] [P01*1*30000*EA*0.42*PN*747355*PD*Circuit Network
[6] [SCH*10000*EA*002*820604
[7] [SCH*20000*EA*002*820709
[8] [CCT*1*30000
[9] [SE*9*159

Figure 3. EDI (X.12). Request for Quotation (line numbers added)

From [15]

[1] This is an RFQ Message * Message Number 159
[3] Seller of item is X, Inc.
[4] Purchaser of item is Y Co.
[5] First Item: 30000 of part 747355 (a Circuit Network) at $0.42/item.
[7] Request that 20000 of the first item be delivered after July 9, 1982.
[8] A total of 30000 items have been requested.
[9] There are 9 lines in this message.

This is the end of message 159.

Figure 4. Approximate English Translation of Request for Quotation

From [15]

Without automated creation of, and response to, electronic messages, there would be little point to EDI. One might as well use plain e-mail.

When EDI messages operate leanly, however, and omit so much information, understanding and processing them becomes a nontrivial exercise. Humans reading paper forms (see Figures 2 and 5) have background knowledge and training that allows them to interpret a message, and know, for example, that the sender of a purchase order (presumably) expects to receive certain goods and to pay for them in a certain way. A trained person could respond to a request for a quotation by ascertaining that the sender should be responded to, checking with the proper source in-house, and forwarding a reply to the sender (say by filling out a form like the one in Figure 2).

Automated processing of an EDI message requires that the computer system be able to store in its own database such important information as the name of the sender, the ID of the message, what the sender wants, and so on. Automated processing also requires that the computer system be able to initiate further action. A very simple example would be a computer system that responds to a purchase order by sending information relevant for the production of the requested goods to the production department, advance notification of the impending shipment of the goods to the carrier, and an invoice, after the goods are shipped, to the buyer. Depending on the delivery terms, the computer might also arrange insurance of the goods while in transit.

All of this, whether done by people or by computer systems, requires (among other things) that:
1. The processor unpacks and understands the meaning of the message to be processed.

2. The processor infers the consequences of the message and acts upon them (once the meaning is understood).

The first item in the foregoing list is called the **meaning problem**, and the second is the **inference problem**. Each will be discussed separately.
The Meaning Problem

The meaning problem, in its basic forms, is also known as the mapping problem. At the most elementary level, a message processor must recognize that a certain field in the message corresponds to, is to be mapped to, a particular field in the recipient’s information system. For example, QUOTE DATE in a quotation form (see Figure 2) might need to be mapped to the QTEDATE field in the QUOTES table of the ACCOUNTS database. The format of the content of the field, a date, will often have to be mapped as well. In general, an organization’s representation of its business in its data-processing system will seldom, if ever, closely match the representation in a received (or sent) business message. The meaning problem is the problem of translating (or mapping) from one representation to another.

It is helpful to think of meaning unfolding as a process (or piece of software) that translates an EDI message into a sentence that can be read by a human or a formula that can be processed by a computer. The following expression represents this process.

\[ MET: u \rightarrow u_{\text{unfolded}} \]  

In the expression, the operator (or process) MET embodies a meaning extraction theory. It transforms the input \( u \) (normally, an EDI message) to the output \( u_{\text{unfolded}} \) (typically, an organization’s representation of the meaning in the received message, \( u \). The output \( u_{\text{unfolded}} \) is the unfolded message corresponding to \( u \). Either or both may be semantically and syntactically complex.

With EDI, meaning unfolding articulates the contents of messages. The messages by themselves are hardly intelligible to the untrained human reader, since only the most basic data are represented, and not the (meaning) relations among them. In meaning unfolding, relations are added to the bare data of an EDI message. For example, an EDI X12 request for quotation (see Figure 3) gives the identities of the prospective buyer and seller, the quantities and types of goods to be purchased, and so on, but does not explicitly state that the buyer is responsible for paying for the goods, and the seller for shipping them. Much information of this sort comes out in the English translation (see Figure 4), which may be thought of as the output of an MET operator, applied by humans. The character and detailed outputs of meaning unfolding will vary depending on whether the subsequent processing is by humans or by machines, but it must occur in some form no matter what. People who process messages received on paper forms must be taught an appropriate meaning extraction theory (MET). When the unfolding is done automatically, the MET must be explicitly programmed. In both cases, it should be emphasized, the MET operator will rely on (a) background knowledge, and (b) the processing organization’s knowledge base (see Figure 1).

An aside to conclude this section.

The existing software that performs the meaning unfolding process—so-called EDI translator software—is widely regarded as inflexible, limited in scope, expensive, and inefficient [26, 27]. Usually, how the mapping is done (i.e., how the MET function works) is idiosyncratic, varying from company to
company. Specialized message forms from any given sender are mapped and translated to the data-processing regime of the receiver. A different mapping and translator configuration is needed for each of a company’s EDI suppliers.

Carefully attending to the meaning unfolding of an EDI message can reduce the magnitude of the translation problem. If a reasonably general theory of what messages can say (a semantic theory) were to hand, then certain efficiencies in implementation would be possible. Having a description of the meaning of an EDI message would be a great help for the programmers who have to develop the translator software. Instead of aiming to map a given message form to a particular firm’s data-processing structure, they could map it to an intermediate representation of high generality and public availability, one with well-known and understood semantics. This will usually be easier than mapping to a specific firm’s data-processing system. Individual firms would have to map from the general intermediate representation to their own data-processing systems, but this is something that need be done only once (and then maintained).

Explicitly representing the meaning of an EDI message does more than just help humans to understand the message. It supports the development of the translator software, and in general of the EDI application software built to process these messages automatically. Full development of these points, however, is beyond the scope of the present paper.

The Inferencing Problem

In addition to the meaning problem, there is also an inferencing problem, that is, the problem of inferring the implications of an EDI message. Some illustrative examples will be discussed below.

Interchange Agreement

EDI messages are only legally binding between partners who have signed a so-called interchange agreement [27]. Without such an agreement, EDI messages have, at best, murky legal implications, and hence may not create obligations for the agents mentioned in the message. Especially for international trade, the interchange agreement has to stipulate under which country’s jurisdiction the agreement resorts. This is very important, because the legal implications of an EDI message can vary from country to country. If an EDI message is legally binding in country A but not in country B (i.e., it is not enforceable in B), then only in country A does the promise stated in the message imply a (legal) obligation.

A simple example will illustrate the point. Suppose that message \( u \) is a purchase order, and that \( r \) honors the purchase order and delivers the goods as requested. Something like the following expression should be inferable:

\[
\phi:
\]

\( 
\text{Buyer B is obligated to pay seller S price P by payment method M for delivery of goods G under terms D.} 
\)
The inference that message \( u \) leads to a condition that is legally binding and obligatory, \( \phi \), is accounted for by the message’s wrapper. A wrapper is a process or piece of software that can make such inferences. Typically independent of the EDI message, it contains information, typically of a general sort, that is not stated explicitly in the message. In other words, the wrapper is the theory that enables the right type of inferences from the data in an EDI message.

The following expression—both analogous to and fitting to Expression 1—represents this process.

\[
\text{CET: } u_{\text{unfolded}} \rightarrow \phi \tag{2}
\]

In the expression, the operator (or process) CET embodies a consequence extraction theory. It transforms the input \( u_{\text{unfolded}} \) (normally, an unfolded EDI message) to the output \( \phi \) (typically, an organization’s representation of a consequence or action resulting from the received message, \( u \), and other information). \( \phi \) is the unwrapping of the unfolded message \( u_{\text{unfolded}}' \).

**Reference to Other Contracts or Procedures**

EDI messages often include references to other contracts of procedures. If the message is to be properly understood, these references must be understood, recognized, and acted upon. For example, Incoterms are sets of clauses for the delivery specifications of purchased goods. An Incoterm determines whether the buyer or the seller is responsible for organizing and paying for transport of the goods from the seller’s place to the buyer’s place (see [7, 9]). According to the Free-on-Board (FOB) Incoterm, the seller is responsible for everything concerning the transport of the goods until they are loaded on the vessel in the port of origin. This Incoterm becomes effective in practice by simply including the term FOB in the purchase order, often under the slot for delivery terms. An inference process is required to determine what obligations are implied by the term FOB in an EDI purchase order. Assume that the meaning of FOB is expressed as follows.

\[\psi:\]

*The seller is obliged to arrange for the shipped-goods export documents, inland transport to the railing of the vessel, and cover all the expenses for this.*

Assume that \( \alpha \) denotes: *Seller S is obligated to buyer B to deliver goods G under delivery terms FOB for price P by payment method M.* Then the wrapper should support the following inference with respect to the obligations implied by the FOB term in the message:

\[
\text{CET: } \alpha \rightarrow \psi \tag{3}
\]

The letter of credit (LC) procedure offers another illustration. For example, in a purchase-order message, the payment procedure can be specified (e.g.,
postpayment, i.e., payment after delivery of the goods, or prepayment, i.e., payment before delivery of the goods). The letter of credit procedure is a more complicated payment procedure in which it is stipulated that a local bank is taking over the risk for the seller of payment by the buyer. The LC procedure requires specific documents, such as a bill of lading. In this case the payment method indicated in the purchase order makes an implicit reference to these other trade documents, which also have to be processed in the right way (see [1, 17]). Again, some form of CET, based on the wrapper (background knowledge plus the organization’s knowledge base; see Figure 1) is needed in order to discern the consequences of messages and act upon them.

**Background Law**

EDI messages should not contradict national or international contract law. For example, under the United Nations Convention on the International Sale of Goods (CISG), if the trading partners in a purchase contract agree that the goods will be delivered under the Free-on-Board (FOB) Incoterm, the seller is responsible for arranging the proper export documents for the shipment of the goods. However, if the seller’s government has banned the transport of this type of goods to the country of the buyer (as the American government used to ban the export of certain types of computers to the former Soviet Union), then the CISG stipulates that in this case the seller is exempted from the FOB obligation to arrange export documents. In other words, the FOB clause in the purchase contract is overruled by the CISG in special situations (see [7, 9]). Assume that the expression Exportban represents this information about the export ban issued by the government on the export of goods of a certain type, such as the goods G mentioned in a message, $u_{\text{unfolded}}$. The blocking of the inference can be stated as follows:

$$\text{CET} : u_{\text{unfolded}} \land \text{Exportban} \vdash \psi$$

(4)

Here $\vdash \psi$ indicates that the inference to $\psi$ no longer holds, due to the extra information about the export ban. This type of reasoning, where conclusions are lost by adding extra information, is called defeasible reasoning (see [15] for early remarks on the use of defeasible reasoning in electronic commerce). One could argue that inferring the legal implications of an EDI message is always defeasible even in a country where the contract is valid, because there might be exceptional circumstances that override them.

All these examples clearly show that it is far from obvious what implications can be inferred from a particular EDI message, hence, what obligations hold for the parties mentioned in an EDI message. The CET inference process, using the wrapper, is needed to extract this information from the EDI message. The CET and the wrapper are typically pieces of software that are independent of the EDI message. They contain background information that is not stated explicitly in the message. This background information could be available via the Web. Prototypes have already been developed that provide online background information. For example, the International Chamber of Commerce (ICC) launched a pilot project called Eterms (http://www.verisign.
Many comments, Message more The meaning which more paper-based The on-line-available Eterms Repository is an on-line database of legal and information-security terms, clauses, model agreements, practice statements, and related materials that serve the electronic commerce community. The purpose of the project is to develop an accessible body of terms, conditions, and related materials that trading partners can use or refer to in establishing electronic commerce relationships. Another pilot project is the on-line-available Interprocs models of trade procedures (http://abduction.euridis. fbk.eur.nl/projects/weboutline/Web.Outline.html). Graphical models of trade procedures downloaded from this site can help merchants to understand which trade procedures apply for their trade. A detailed description of this project will be found in [17].

Practically speaking, not every aspect of wrapping can be made explicit. Many of the wrapper examples mentioned above also exist in the traditional paper-based document environment. In paper contracts these issues are not addressed explicitly either. However, in the new electronic environment of electronic commerce, there is incentive to make as much as possible of this implicit information explicit, and thus available for automated processing. A complete wrapper theory that would answer every question is probably not feasible. However, developing the wrapper theory is an incremental process. The broader the theory, the more likely that automated processing can use it.

**Summary So Far**

Message unfolding is the generating of a semantic model of an EDI message. Inevitably this requires going to a more detailed level of explanation than is normally the case. It is the wrapping that requires (and rewards) going to a more abstract level, allowing a message to get different interpretations depending on the legal system and other relevant background information by which it is processed. The difference between meaning unfolding and meaning unwrapping can be illustrated as follows. Meaning unfolding of an EDI purchase order is adding, for example, the information that the buyer is responsible for payment of the goods. However, this responsibility only implies a legal obligation for the buyer if the purchase order is legally binding, which depends on the interchange agreement. And this legal inferencing has its basis in the wrapping of the EDI message. The basic idea is that the obligations do not automatically follow from the meaning of an EDI message. They follow from the message and the wrapper. The need for meaning unfolding and meaning unwrapping of EDI messages also explains the observation that electronic commerce is much more than just exchanging EDI messages.

To sum up, EDI messages are lean in the sense that they tend to express no more than a minimal, or core, meaning. The larger, or extended, meaning of the message can only be determined in light of the governing wrapper theory, which includes the interchange agreement and relevant background laws. Thus, to interpret a message, a recipient must first unfold its core meaning, and subsequently use the wrapper theory to make action-relevant inferences. The following points summarize the argument so far.

1. The EDI (and other business-to-business) messaging schema in
Figure 1 describes the essential context of EDI messaging (for present purposes). The focus here is on the problem faced by \( r \), the recipient of a message, \( u \), in interpreting \( u \) and taking appropriate action. What is said will apply straightforwardly to the problems faced by \( s \), the sender of the message, who must figure out what the message should say.

2. Figures 2–5 present representative business messages, either as EDI messages or as paper-based forms.

3. Recipient \( r \) has available contextual information that it can use in interpreting and responding to the message \( u \). This information is called the \textit{wrapper} (for \( r \)). In terms of the basic messaging schema in Figure 1, \( r \)'s wrapper consists of the background knowledge (including any interchange agreement) and \( r \)'s knowledge base.

4. Whether \( r \) is a person or a computer system, its processing of the message has two key requirements:

   (a) The meaning of the message to be processed must be unpacked and understood by the processor.

   This is called the \textit{meaning} problem. It is handled by applying a meaning extraction theory (MET) to the message, thereby yielding a (meaning) unfolded expression. In symbols:

   \[ \text{MET} : u \mapsto u_{\text{unfolded}} \]  

   This process is called \textit{unfolding}.

   (b) The consequences of the message (once its meaning is made explicit and understood) must be inferred by the processor and acted upon.

   This is called the \textit{inference problem}. It is handled by applying a consequence extraction theory (CET) to the unfolded message, thereby yielding a collection of direct consequences, \( u_{\text{cons}} \) (think of it as a conjunction of possibly many expressions), which \( r \)'s processing system can record and directly act upon. In symbols:

   \[ \text{CET} : u_{\text{unfolded}} \mapsto u_{\text{cons}} \]  

   This process is called \textit{unwrapping}.

5. Both the MET and the CET inevitably draw upon the wrapper. This is true whether the processing of \( u \) (the message) is automated or done by humans. Context must be added in order to interpret and act upon a received message.

6. The architecture described in the preceding points allows—and actual business practice (including EDI messaging systems)—realizes a certain leanness in the design of messages. The message (see Figures 2–5) is abstracted from its underlying meaning (see Figure 8). This
permits message formats to remain constant, while the meanings they convey are changed by context.

This completes the overview of the points that had to be emphasized. The discussion will now turn to the task of providing more concrete illustrations.

**Meaning Unfolding**

First it is necessary to make two distinctions. The logical theory offered below will be sensitive to, and make use of, both of them.

First, there are two sorts of meaning: *core meaning* and *extended meaning*. This distinction, alluded to above, is similar to the distinction in speech-act theory between locutionary and illocutionary acts. Speech-act theorists differentiate the meaning of a sentence (locutionary act) and the meaning of the speaker of the sentence (illocutionary act), even if the two meanings coincide. For example, the sentence meaning of “Do you know what time it is?” (an inquiry of the addressee whether the addressee is aware of the hour) is different from the usual speaker meaning (a request to be told the time). Distinctions of this kind are recognized in ordinary language when one says something like, “What she said was X, but what she meant was Y.”

As noted above, EDI messages have certain core meanings (in virtue of the standards to which they conform) as well as additional meanings that are unpacked (unwrapped and unfolded) in light of the wrapper (governing interchange agreement, etc.).

The second distinction is a three-fold one. For statements, or messages, three levels of representation are distinguished:

1. Informal: Typically in natural language.
   Example:
   
   “All politicians are liars.”

2. Formal: Syntactically formal, but not fully logical (e.g., without formal rules of inference or a formal semantics).
   Example (for representing “All politicians are liars,” i.e., (7) → (8)):

   \[(politician \text{ and liar})(x)\]

3. Logical: Syntactically and semantically formal, that is, in a logic and with a formal semantics.
   Example (for representing “All politicians are liars,” i.e., (7) → (9) and (8) → (9)):

   \[\forall x \,(P(x) \rightarrow L(x))\]

Now, if machines are to process messages effectively, the messages cannot be expressed informally, as in Expression 7. Instead, formal expressions that use an appropriate and convenient syntax are required, as in Expression 8. EDI messages, when they do not include free text, are formal, but not logical. Important for the success of a formal messaging syntax, as in Expression 8, is
a theory of (semantics for) what the messaging syntax means. Expression 9 provides this (in outline) for Expression 8.

Note further that Expression 9 is really an unfolding of the meaning of Expression 8. It articulates, more precisely and in more detail, what Expression 8 means. Why? Because Expression 8 is (we assume) expressed in a formal syntax without accompanying logic or semantics, while Expression 9 is represented in logic and has (we assume) an appropriate accompanying semantics (i.e., the standard one). Of course, it would be possible to unfold and articulate the meaning of expressions like (8) without advertizing to formal logic, but there is little reason to do so. Formal logic is well developed theoretically and presents something of a gold standard for formal clarity. That is fundamentally why it is used for the purposes of this discussion.

The point is not that formal logic expressions like Expression 9 should be used for business-to-business communication instead of syntactically convenient expressions like Expression (8). Instead, the point is that a theory of unfolding—with a semantics, as in Expression 9—is valuable for understanding how to design and validate effective automation systems for electronic commerce, and that formal logic is an appropriate vehicle for expressing such a theory. For example, a proper logic provides an automatable way of exploring the inferences that can or cannot be made from a collection of statements. Especially for the complex and subtle reasoning needed to support electronic commerce (e.g., reasoning about commitments and obligations), this should be valuable both for specifying and validating software systems (working, e.g., with syntactically formal EDI expressions) and for facilitating the construction (dare we hope automatically?) of interchange agreements and other wrapping elements.

**Unfolding a Simple Promise**

Meaning unfolding is the articulation of the basic semantics for an expression. Using the example of promising, meaning unfolding can be thought of as follows: Let us say that at the informal level of representation the expression reads:

At time \( t \), \( s \) promises \( r \) that \( s \) will deliver to \( r \) goods \( g \) within 30 days. \( \text{(10)} \)

Using an EDI-like message notation, this might be expressed formally (but not logically):

1. \( \text{promise} : 12345 \)
2. \( \text{date-time} : 1999-09-23 \)
3. \( \text{from} : s \)
4. \( \text{to} : r \)
5. \( \text{deliver} \)
   
   \( (a) \) \( \text{goods} : g \)
   \( (b) \) \( \text{to} : r \)
   \( (c) \) \( \text{by} : s \)
   \( (d) \) \( \text{date-time} : 1999-09-23 + \text{day}(1999-09-23 + 30) \) \( \text{(11)} \)

Thus, Expression 11 should be regarded as an instance of \( u \) in Figure 1.
At level 3, logical representation, this promise can be represented as fol-

\(\text{promise}(12345) \land \text{Speaker}(12345, s) \land \text{Addressee}(12345, r) \land \)

\(\text{Cul}(12345, 1999-09-23) \land \Box (T \rightarrow (K(12345) \leftrightarrow \text{deliver}(e) \land)

\text{Agent}(e, s) \land \text{Benefactive}(e, r) \land \text{Sake}(e, 12345) \land \text{Theme}(e, g) \land \)

\(\text{Cul}(e, t) \land \leq (t, +(1999-09-23, \text{day}(1999-09-23 + 30))))\) (12)

Comments and points arising:

1. Expression 12 reads roughly as “12345 is a promise by s to r, occur-
ring on September 23, 1999. The promise 12345 concerns a delivery
event e. Necessarily, this promise is kept if and only if s effects a
delivery to r of g, and this is for the sake of the promise 12345. In
addition, the delivery e must occur (for the promise 12345 to be kept)
within 30 days of September 23, 1999.”

2. This reading is also (one assumes) a correct interpretation of the
EDI-like message Expression 11, and of the original ordinary
language-like utterance, Expression 10.

3. \(\Box\) in Expression 12 is a modal logic necessity operator. For any
sentence \(P\), read \(\Box P\) as “Necessarily, \(P\)” or “It is necessary that \(P\).”
This necessity operator is required for technical reasons, which for
present purposes are elided. It is generally safe simply to drop the
operator \(\Box\). Doing this yields the extensional approximation, and for
practical purposes the approximation suffices.

4. Unlike Expression 11, Expression 12 is logical and fully formal. It is
also the unfolded version of Expression 11. Note in this respect how
Expression 12 makes explicit so much that would otherwise have to
be read into or assumed about Expressions 10 and 11.

5. The predicates in Expression 12 all happen to be from a (here
unspecified) controlled vocabulary, and would be used in unfolding
very many types of primary messages.

6. There is nothing special about promising that limits the scope of this
approach to representing messages. The move in evidence here
generalizes to the other illocutionary forces, such as asserting,
requesting, and declaring. Although further details are required, the
analysis is remarkably simple. Even so, it is able to capture the core
logical behavior of various illocutionary forces. See [10, 11, 12, 13, 14].

7. T, appearing in Expression 12, is shorthand for any logical tautology.
It always evaluates to true. Thus anything conditioned on T is only
vacuously conditioned. If someone says that he will go to the store
whether or not it rains, his going is but vacuously conditioned on the
weather.

8. Note that the meaning, as analyzed, implies nothing about whether
certain actions are obligated by a promise. Promising is one thing,
keeping a promise is another. This accords well with the leanness of
EDI messages, noted and discussed above. For example, it standard-
ly belongs to the meaning of a purchase order that the seller delivers
goods to the buyer and the buyer pays the seller, but these acts are
only obligatory for these parties if additional assumptions are made.
Making these assumptions is normally part of what the interchange
agreement (the wrapper) is about. In this respect the analysis here
deviates from the standard view, or central dogma, of speech acts,
exemplified in the work of Searle and Vanderveken [23, 24, 25, 31]
among others. Searle and Vanderveken assume that a speech act of
promising automatically implies an obligation to fulfill the promise.
However, this implication critically depends on the context. It might
hold for some contexts and not for others. This inference is per-
formed by using the wrapper, which will be discussed below. See
[14] for a more detailed discussion of this point.

Unfolding a Simple Purchase Order

Figures 6 and 7 show, respectively, a simple example of a specific purchase
order and a representation of the same purchase order in an EDI-like (struc-
tured, not logical) syntax. The names of the data items have been kept short,
and the number of data items few, for the sake of convenience. The example is
rich enough to bear the points to be made, and sparse enough to avoid bur-
dening the reader with unnecessary detail. Actual purchase orders, and the
EDI protocols for them, are, of course, much more complex. Full treatment of
such examples is beyond the scope and purposes of this paper.

Figure 8 displays the results (the \( u_{\text{unfolded}} \)) of using lean-event semantics to
apply a reasonable meaning extraction theory to the \( u \) of Figures 6 and 7.
Although there is very much to say about this example, the discussion here is
limited to a few remarks that make the example accessible and clarify its con-
nection with the points made earlier.

1. The expression in Figure 8 is a fully formal, logical \textit{model} (and hence,
approximation) of what would typically be meant by the purchase
order shown in Figure 6.

2. As before:
   (a) Predicates beginning with a capital letter (\textit{Speaker, Theme, Comc,}
       etc.) are generic reserved words in lean-event semantics;
   (b) Predicates in lower case (e.g., \textit{deliver, pay}) are (here) verbs whose
       meaning is specified independently (more on this below); and
   (c) Predicates in all upper case (e.g., \textit{FOB, TERMS}) are domain-
specific.

3. The expression in Figure 8 is a single logical conjunction of the three
main blocks of code.

4. The gist of the first block is that this is a purchase order from \( s \) to \( r \)
happening ("culminating," \textit{Cul}) on 1999-11-19, regarding (having
\textit{Theme} of) three events: \( e1, e2, e3 \).
Figure 6. A Simple Purchase Order

5. The gist of the second block is that this purchase order is making a request (of r by s). The request is honored (H) if and only if two shipping events occur, e1 and e2. Descriptions of these events follow and match to the original purchase order. Note that in both cases, the shipping events are (requested to) begin ("commencing," Conc) during the period indicated. Nothing is said about when delivery should occur.

6. The gist of the third block is that if the aforesaid request is in fact honored, then this purchase order is also making a promise. The
1. purchase-order : 54321  
2. date : 1999-11-19  
3. from : s  
4. to : r  
5. ship-to : an-address  
6. terms : 30-days-net  
7. ship-week-of : 1999-12-03  
8. FOB : customer  
9. grand-total : 276.03  
10. deliver  
   (a) goods : catid-32-9  
   (b) unit: box  
11. sell  
   (a) goods : catid-35-9  
   (b) unit: dozen  
   (c) description : “no. 2 pencil”  
   (d) quantity : 3  
   (e) unit-price : 1.99  
   (f) line-total : 5.97

---

Figure 7. EDI-like Representation of the Simple Purchase Order in Figure 6

```
purchase-order(54321) ∧ Speaker(54321, s) ∧ Addressee(54321, r) ∧ Cu1(54321, 1999-11-19) ∧
□ (T → (H(54321) ↔
((ship(e1) ∧ Agent(e1, r) ∧ Benefactive(e1, s) ∧ to(e1, an-address) ∧ Theme(e1, catid-32-9) ∧ Sake(e1, 54321) ∧ unit(e1, catid-32-9, box) ∧ description (catid-32-9, “copier paper”) ∧ quantity(e1, catid-32-9, 6) ∧ Com(e1, t1) ∧ FOB(e1, customer) ∧ during(t1, week-of(1999-12-03))) ∧
(ship(e2) ∧ Agent(e2, r) ∧ Benefactive(e2, s) ∧ to(e2, an-address) ∧ Theme(e2, catid-35-9) ∧ Sake(e2, 54321) ∧ unit(e2, catid-35-9, dozen) ∧ description (catid-35-9, “no. 2 pencil”) ∧ quantity(e2, catid-35-9, 3) ∧ Com(e2, t2) ∧ FOB(e2, customer) ∧ during(t2, week-of(1999-12-03))) ∧)
∧
(H(54321) → K(54321) ↔
(pay(e3) ∧ Agent(e3, r) ∧ Benefactive(e3, r) ∧ Sake(e3, [e1, e2]) ∧
Theme(e3, p3) ∧ unit(e3, p3, $) ∧ quantity(e3, p3, q) ∧ unitprice(e1, catid-32-9, p1) ∧ unitprice(e2, catid-35-9, p2) ∧ unit(e1, p1, $) ∧ unit(e2, p2, $) ∧ quantity(e1, p1, 45.01) ∧ quantity(e2, p3, 1.99) ∧ = (p3, +( *(45.01, 6), *(1.99, 3))) ∧ Cu1(e3, t3) ∧ TERMS(e3, 30-days-net)))
)
```

Figure 8. Meaning Unfolding, Via Lean-Event Semantics, of the Simple Purchase Order Shown in Figures 6 and 7

promise is kept (K) if and only if e3 is a paying event with the properties described.

7. Note that the unfolded expression is fully logical and supports inferencing. For example, suppose the purchase order is issued and in fact r honors it fully (ships the goods as, when, and where described). Suppose further that s fails to pay r as described. It follows, logically, that the promise associated with the purchase order, made by s, has not been kept.

8. Suppose, indeed, that the promise has been broken. In order to extract these consequences, a CET and a look at the wrapper are required, as will be discussed in the next section.
An Aside on Mapping to the World

No semantics can ever be entirely formal. At some point one needs to map between the formal symbols used in the formal language and the system that language is to model. The trick is to give this mapping at a very basic level, and then to rely as much as possible on the composition of symbols to provide additional meaning. With the logical semantics presented here (see Figure 8), it is necessary to provide a basic mapping for: terms (s, r, etc.), functions (e.g., \textit{week-of}), and predicates (e.g., \textit{ship, pay}). Terms are rather straightforward—one makes a catalog, a sort of table, in which the referring expression is mapped to what it refers to. For example, \textit{s} might be the customer number of the firm Nadir, Inc. Functions are also straightforward—one either maps them, much as terms are mapped, or defines them by composition from more primitive functions.

Predicates, as was noted above, are of three kinds:

1. Specific for lean-event semantics (e.g., \textit{Theme, Comc, Unit})
2. Generic (e.g., \textit{ship, pay})
3. Domain-specific (e.g., \textit{FOB, TERMS})

The specific predicates should be few in number and can be defined (mapped) explicitly, as in the discussion on Incoterms. Generic predicates can and should be mapped to a clear, broadly accepted, public specification. Language having the importance it does, such specifications are certainly available. WordNet, an electronic lexicon produced over a number of years at Princeton University, is an excellent example [6]. Consider what WordNet has to say about the three generic (verb) predicates in Figure 8: \textit{purchase-order, ship, pay}.

WordNet recognizes \textit{purchase order} as a noun but not as a verb. Nonetheless, what it says is useful. The following passage, as well as all the subsequently quoted passages, is from WordNet 1.6.

The noun purchase order has 1 sense (no senses from tagged texts) 1.
(04902219) \textit{noun.communication} order\#7, purchase\ order\#1—(a commercial document used to request someone to supply something in return for payment; “IBM received an order for a hundred computers”)

Note that a purchase order here (and in Figure 8) involves both a request for something and a (promised) payment for it.

WordNet 1.6 recognizes \textit{ship} as a verb for which there is exactly one sense: to transport commercially.

1 sense of ship
Sense 1
(01328437) \textit{verb.motion} transport\#4, send\#4, ship\#1—
(transport commercially)

\[\Rightarrow (01328337) \textit{verb.motion} bargel\#2—
(transport by barge on a body of water)
\[\Rightarrow (01331285) \textit{verb.motion} dispatch\#1, despatch\#1, send\ off\#1—(send off promptly)
\[\Rightarrow (01331167) \textit{verb.motion} bundle\ off\#1—
(send off unceremoniously)\]
More refined meanings are available with different verbs. Thus the ship of Figure 8 may be tentatively identified with ship\#1 of WordNet. Finally, WordNet 1.6 recognizes 11 distinct meanings for pay as a verb.

The verb pay has 11 senses (first 11 from tagged texts)

1. {01540968} <verb.possession> pay\#1—
   (give money in exchange for goods or services; “I paid four dollars for this sandwich”; “Pay the waitress, please”)

2. {00718708} <verb.communication> give7#5, pay7#2—
   (convey, as of a compliment, regards, attention, etc.; bestow; “Don’t pay him any mind”; “give the orders”; “Give him my best regards”; “pay attention”)

3. {01541614} <verb.possession> pay up\#1, ante up\#1, pay4#3—
   (cancel or discharge a debt; “pay up, please!”)

4. {01542031} <verb.possession> pay2#4, pay off4#4, make up3, compensate2#4—
   (do or give something to somebody in return; “Does she pay you for the work you are doing?”)

5. {01695538} <verb.social> pay5—
   (render; “pay a visit”; “pay a call”)

6. {00500874} <verb.cognition> pay3#6—
   (bear (a cost or penalty), in recompense for some action; “You’ll pay for this!”; “She had to pay the penalty for speaking out rashly”; “You’ll pay for this opinion later”)

7. {01566906} <verb.possession> yield10, pay7#7, bear7#8—
   (bring in; as of investments; “interest-bearing accounts”; “How much does this savings certificate pay annually?”)

8. {01869530} <verb.stative> pay8—
   (be worth it; “It pays to go through the trouble”)

9. {00496485} <verb.cognition> give2#10, pay9#9, devote2#—
   (as in the expressions “give thought to”; “give priority to”, etc.)

10. {01541783} <verb.possession> pay4#10—
    (discharge or settle; “pay a debt”; “pay an obligation”)

11. {01600647} <verb.possession> pay3#11—
    (make a compensation for; “a favor that cannot be paid back”)

Pretty straightforwardly, pay can be identified with pay\#1.

These verb mappings to WordNet happen to be unproblematic. Even so, a degree of clarification has been gained. More important, nothing like this is available without some form of explicit meaning unfolding. The discussion now turns to how the wrapper can be used to extract further consequences from business messages.
Unwrapping

For reasons mentioned above, the illocutionary, deontic, and other consequences of an EDI message typically have to be inferred. These inferences may be defeasible [18, 21, 22]. Think of modeling the legal implications of a contract. Basically these are not represented in the EDI message, but can be inferred from the EDI message(s) and a background theory (which belongs to the wrapper). A typical example is to infer whether a promise really implies an obligation, or in more legal terms, to infer the legal status of an EDI message. All of this may vary from country to country, since the wrapper is country-dependent. These are large and important issues, and they cannot be resolved in a short paper, although it is possible to make a start.

In the authors’ view, the necessary inferences attending to EDI messages can be achieved through a system of lean messaging with subsequent unfolding and unwrapping, relying on the wrapper. There are good reasons to mirror this strategy in new protocols and in the development of a formal semantics for EDI, and this paper offers (a sketch of) an account of how to represent the lean semantics (the unfolding) of an EDI message. The aim now is to indicate how to formalize part of the wrapping so that required additional inferences—the unwrapping and extraction of consequences—can be made automatically. Directed obligation will be the focus, by way of example.

Modeling directed obligation is particularly interesting and challenging. Standard deontic logics are about obligations tout court. For business and other purposes, there is need for a concept, and thus a formalization, of directed obligation in which one agent has an obligation to another agent to do something. In the case of our purchase order, if r delivers, then s not only has to (is obligated to) pay for the goods, but has to pay r in particular. Investigation of how to model this began only recently, and the initial results were not fully satisfactory (see [8]). Tan’s group, however, has developed an analysis and a logic for directed obligation (see [29, 30]). The central formula

\[ O^d_i (εα) \equiv (¬εα ⇒ S \; P(ε(α S i))) \]  

(13)

represents their analysis of directed obligation and should be read as “Under legal system S, i is obligated to j that i brings about [ε] state α in the world means if i fails to bring about α then under system S, j would be permitted [P] to bring legal action [εα] against i.” By way of explanation, the operators \( O^d \), \( ε \), and \( P \) are modal operators. The operator \( O^d_i \) is a deontic that expresses that agent i is obligated to j under legal system S to bring something about. The operator \( ε \) is a so-called action operator that expresses that agent \( P \) brings something about. The operator \( P \) is again a deontic operator for the axioms of standard deontic logic are assumed (for technical details, see [29, 30]). The operator \( O^d_i \) is defined in terms of the two other operators \( ε \) and \( P \). There are two fundamental differences between \( O^d_i \) and a standard deontic logic operator O. First, \( O^d_i \) represents a directed obligation between two specific agents i and j, whereas O is completely anonymous. Second, \( O^d_i \) defines in purely objective and operational terms what will happen if the obligation is
violated. For example, if a buyer has an obligation to the seller to bring about payment of the delivered goods, then the seller has permission to take legal action against the buyer if it does not pay. Symbolically, this gives:

\[ b \circ s (b, s, \text{paying}(e)) \overset{\text{def}}{=} (\neg b, \text{paying}(e) \Rightarrow s, P(s, e, la, s)) \]  

where \( b \) is the buyer and \( s \) the seller.

The operational character of this definition does not simply give a detailed explanation of what it means to have a contractual obligation between two trading partners. It also makes the notion of directed obligation applicable to artificial agents, because it is only the operational behavior of the agent that matters and not its internal mental state, which artificial agents probably lack.

With this definition (and the attendant logic), it is possible to write bridge laws or meaning postulates or Protokolssätze that connect the language of messaging with the language of directed obligation. Here are two examples:

\[ \forall e, e', s, r \]  
\[ (\text{purchase-order}(e) \land \text{Speaker}(e, s) \lor \text{Addressee}(e, r)) \land \text{accept}(e') \land \text{Agent}(e', r) \land \text{Theme}(e', e)) \]  
\[ \rightarrow (\neg s, H(e) \Rightarrow s, P(s, e, la, s)) \]  

If you get a PO and accept it (i.e., you send a message in which you say \text{accept}(e') \land \text{Agent}(e', r) \land \text{Theme}(e', e)), if you don't honor it (deliver the goods), then the sender of the PO can initiate legal action against you.

\[ \forall e, s, r \]  
\[ (\text{purchase-order}(e) \land \text{Speaker}(e, s) \land \text{Addressee}(e, r) \land \text{H}(e)) \rightarrow (\neg s, K(e) \Rightarrow s, P(s, e, las, s)) \]  

If you send a PO and it is honored (the goods are delivered) and you don’t pay, then the receiver of the PO can bring legal action against you.

Expression (16) shows clearly the technical difference between meaning unfolding and wrapping. The predicate \( K(e) \) is an essential part of the unfolding of the purchase order (see Figure 8), but in this unfolding nothing is said about obligations. Nothing in Figure 8 says that \( K(e) \) is obligated! To express this deontic aspect, it is necessary to add the definition of directed obligation \((\neg s, K(e) \Rightarrow s, P(s, e, las, s))\). Note that this adding of rules regarding obligations is critically dependent on whether the purchase order is legally binding according to an interchange agreement.

Comments:

1. Lean-events semantics theory has paid off handsomely here. Notice that the bridge laws ((15) and (16)) are completely general and use only the controlled vocabulary (verb predicates, thematic role predicates (from the controlled vocabulary of lean-event semantics), and predicates from the logic of directed obligation).

2. The illocutionary auxiliary predicates (here \( H \) for “is honored” and \( K \) for “is kept”) of Kimbrough’s lean-events theory of speech acts have a natural and essential place in the example bridge laws, allowing them to be general.
3. The preceding point leads one to wonder whether lean-events semantics theory might be used to analyze further the logic of directed obligation. Probably, but that is a subject for another paper.

4. Much remains to be done if wrappers for electronic commerce are to be substantially formalized and given a full semantics. This could be a subject for many subsequent papers, but there are apparently no fundamental impediments to completing such a program of research.

**Conclusion**

This paper has presented an argument for a way of thinking about how transactions work in electronic commerce (lean messaging in the context of wrappers, meaning unfolding, representation with lean-events semantics, unwrapping to extract inferences, directed obligation theory, etc.). It was observed that under existing arrangements, EDI messaging is lean, messages are unfolded (in industry lingo, “mapped” to the receiver’s data-processing system), and then unwrapped (in industry lingo, “processed”). This arrangement, so far as it goes, seems right, even inevitable—messages carry news, wrappers hold context.

In addition to this basic framework, the paper has presented a demonstration of how two particular theories—Kimbrough’s lean-events semantics, and Tan and Thoen’s theory of directed obligation—can be used to account for unfolding and unwrapping. These two theories bring considerable virtues to the table. They are both logical theories and come with proof theories and meaningful semantics. Both, in virtue of being broadly logical, can be mapped to ordinary language in a foundational way, per the discussion of mapping to the world. And they can be made to work together, as was shown here with the example of bridge laws.

Significantly, however, both theories—lean events and directed obligation—are of recent origin and still under development. Neither may turn out to be the best theory in its class for modeling messages or wrappers. Only future research can settle the issue. That there is an issue, and that it is clearly articulated, is the main contribution of this paper.

**NOTES**

1. Regarding computer-to-computer exchange of structured information, this can serve as a rough definition of EDI (electronic data interchange).

2. To the authors’ knowledge, Ken Steel was the first to make this point forcefully and in print [28]. It has been seized upon by the XML community. Steel did not offer a comprehensive semantics for the intermediate form, nor has the XML community demonstrated how a DTD can perform such a function.

3. Of course, this is somewhat idealized. One of the purposes of interchange agreements is to specify what the standards leave vague.
REFERENCES


For Steven O. Kimbrough’s biographical information, see the Guest Editors’ introduction.

For Yao-Hua Tan’s biographical information, see the paper by Bons et al. in this issue.