

EDI, XML, and the Transparency Problem in Electronic Commerce*

Steven O. Kimbrough
University of Pennsylvania

February 5, 2000

Contents

1	EDI and the Transparency Problem	1
2	XML's Pertinent Virtues and Limitations	3
3	Communications Requisites	5
4	microFLBC and the Transparency Problem	12
4.1	Representing a simple promise	15
4.2	Representation of a Simple Purchase Order	21
4.3	Mapping to the world	23
5	Back to XML	26
6	Conclusion	31
	References	31

*File: flbc-transparency.tex. This material is based upon work supported by, or in part by, DARPA contract DASW01 97 K 0007.

Abstract

Standard (that is, long-standing and currently much in use) EDI protocols (including the X12 and EDIFACT series) have repeatedly been criticized for poor design, confusing or absent semantics, and much else. Most of these criticisms are indeed on the mark. The main conclusions that the critics have drawn are also correct: business to business e-commerce is expensive and difficult to set up and maintain, because of shortcomings in the design concepts underlying standard EDI. Something must be done, but what?

Central to the problem is the fundamental question of semantic transparency: When A sends B a message, how does B's machine know what the message is about, what it means? Given proper standards, message meanings are determined and computers can be programmed to act appropriately to the intended message meanings. The complaint against EDI has been that proper standards cannot be made because of the misguided way in which the EDI standards are designed.

Proponents of XML have been touting XML's strengths and claiming that they overcome, or can overcome, the semantic transparency problem in e-commerce. In support of this claim, proponents point to the DTDs the any XML/EDI solution would use. The claim is that semantic transparency is/can be achieved through the DTDs.

In this paper I argue that indeed the DTD mechanism offers a kind of progress on the semantic transparency problem, but that it cannot provide anything approaching a complete solution. While XML+DTDs is indeed a very promising vehicle for structuring and transporting messages for business-to-business commerce, it is not itself a semantic theory of what those messages say. We need the semantic theory. Once we have that, XML can be used to embody it for applications.

Drawing on previous work, I will present the elements of my formal semantic theory for business messaging (the "lean events theory"). With examples from this theory before us, we can get a more proper view of the semantic transparency problem (aka: the spanning problem). This is not a problem that can be made to go away entirely, but we can live with it and do commerce.

1 EDI and the Transparency Problem

The standard EDI protocols (including the X12 and EDIFACT series) are of long-standing and effective use.¹ Yet, they have repeatedly been criticized for poor design, confusing or absent semantics, and much else.² Many of these criticisms are indeed on the mark. The main conclusions that the critics have drawn are also correct: business-to-business e-commerce is expensive and difficult to set up and maintain, because of shortcomings in the design concepts underlying standard EDI.

The problem is particularly acute for SMEs (small and medium-sized enterprises), which all too often find the cost of initial setup prohibitive. This cost includes substantial investment in legal services, management and staff time, and software. In the world of EDI, this is often referred to as the *first trade problem*. Once an EDI system is up and running smoothly, the incremental costs can be quite low and the incremental benefits very high. The problem is getting the *first* messages sent and working properly. That typically involves a large fixed cost.³

And, the first trade problem is hardly peculiar to the world of EDI. More broadly, it is recognized that there are important conceptual and technical issues to be addressed if the common vision—of millions of artificial agents cruising the Internet and doing deals for their owners in *ad hoc* and opportunistic ways—is to be realized. This is sometimes called the *spontaneity problem* [4]; it is a more challenging generalization of the EDI first trade problem.

Something needs to be done to address these problems—first trade, spontaneity, etc.—but what?

A central issue—and the focus of this paper—is the fundamental question of *semantic transparency*: When S (speaker) sends A (addressee) a message, how does A's machine (or indeed A itself) know what the message is about, what it means? Or, given that A needs to understand a particular message, how can this be achieved at low cost and hence in a maximally-automated fashion?

Given proper standards, message meanings are determined and computers

¹See, e.g., [6, 7, 11] for valuable general introductions to electronic data interchange (EDI).

²For a sample of the criticisms, many from a basically friendly perspective, see: [1, 3, 4, 11, 16, 18, 21, 24, 32, 36, 37].

³On the EDI first-trade problem see [23, 39, 38] and papers throughout [1].

can be programmed to act appropriately to the intended message meanings. This is something that happens millions of times daily. The rap against EDI has been that proper standards have not in fact been created, and indeed cannot be made because of the misguided way in which the EDI standards are designed. The worry associated with the spontaneity problem is that we do not know how to design a communication regime that will support the vision.

Into this unresolved situation comes XML. Proponents of XML have been touting XML's strengths and claiming that XML overcomes, or can overcome, the semantic transparency problem in general, and the e-commerce first trade problem in particular. It would serve no constructive purpose to rehearse the particular misstatements and exaggerations that have been committed by the advocates of XML. Rather, the question is—regardless of what anyone has claimed—What can XML do to address the transparency problem in e-commerce?⁴ The aims of this paper are to answer this question (regarding XML's capabilities), and—since the answer will be a negative one, albeit qualified—to state positively something on how the transparency problem may be solved. To this end, the paper is organized as follows.

In §2 I briefly review why XML (or anything like it) might be—and has been—thought capable of contributing to solving the transparency problem. I assume a passing familiarity with XML on the reader's part. While XML has considerable virtues, I shall argue that it cannot in any very meaningful sense solve the transparency problem. §3 is devoted to the more fundamental discussion of how meaning is communicated between machines (and indeed people). There is nothing irredeemably mysterious here. Once we remind ourselves how it works, we see better why XML cannot solve the transparency problem and we can see better how the problem can be addressed. §§4–5 carry the main burden of the argument. In §4, I provide a formal grammar for microFLBC, a simple but surprisingly powerful language. microFLBC, I shall argue, can be used to provide a genuine semantics for business messaging and can provide a foundation for addressing the transparency problem. In §5, we return to XML and I describe through an example how XML, in conjunction with microFLBC, can yield actual progress on the transparency problem. §6 concludes, and positions the small degree of progress evidenced here in the context of the much larger problem.

⁴The curious reader, however, can look to [5], [9], and [40] for examples of honest, perhaps too exuberant, overstatement.

Now to the details.

2 XML's Pertinent Virtues and Limitations

HTML's simplicity and beauty of design doubtless contributed enormously to the extraordinarily rapid acceptance of the World Wide Web. Documents marked up in HTML may easily be stored, served from, and read on essentially any widely used computer, using any standard display device. Not only does HTML support a quite usable set of document display features (lists, tables, images, etc.), but it has built-in mechanisms for enabling hypertext linking and email addressing (among many other useful features). Add the fact that the elements of HTML can be learned and applied in just a few hours, and we can begin to appreciate a truly elegant and effective technical design.

Given all these virtues, what's wrong with HTML and why is there so much excitement about replacing it with XML? Why not, as with other technologies, proceed with incremental refinement? Quite simply, the demand for new features far outstrips the ability of any standards-setting body to keep up. Further, if multiple standards proliferate the World Wide Web will lose one of its greatest strengths: universal access. Even Microsoft supports XML strongly, thereby eschewing any attempt to dictate a universal standard for HTML.

But despair in keeping up with demand is not the only major source of criticism of HTML. Perhaps even more important is HTML's nearly exclusive orientation towards display of documents. HTML tags—the markup supported by HTML—are nearly entirely content neutral. It is both their main virtue and their main weakness that they apply to all documents, regardless of content, regardless of what the documents are about. A numbered list is the same in a public presentation, a private memo, or an invoice. This facilitates display by arbitrary browsers, but it hinders processing of the underlying document by specialized applications. It is just this latter purpose that must be served for electronic commerce. As envisioned by the many proponents of XML, electronic data interchange (EDI) messages will migrate away from the present standards (e.g., X12 and EDIFACT) and be expressed in XML. But the point of EDI is computer-to-computer exchange of information. The messages must be generated and processed as automatically as possible. This, in turn, requires that the structure of a message

substantially informs the processing of the message. A list of items cannot, as in HTML, be *merely* a list; we need to know, e.g., whether it is a list of items to be sold or items to be bought.

XML—as well as SGML, its larger, more general but too unwieldy for the Web parent—has much to offer by way of addressing these two problems. For present purposes, it is fair to say that there are two key moves or ideas present in XML (and SGML). First, the philosophy is to use markup (the tags) for specifying content. Presentation, or display, instructions are specified in a separate, general accompanying document, called the *stylesheet*. This addresses our second problem, above. Documents are marked for processing purposes. Stylesheets, which apply to all documents in a given class and which can be referenced in particular documents, may be used by browsers in presenting the documents on screen or in print.

The second key move addresses our first problem, above, that of too many features and forms. If documents are to be marked for processing purposes, are not those purposes manifold? Indeed they are. How, one then wonders, does this increasing of the uses for marked up documents cohere with the need to reduce the onrush of features in our markup language? Is there something about XML (SGML) that circumvents the need in HTML to add new features? Indeed there is: the DTD (document type definition). The purpose of a DTD is to define correct usage of the tags. HTML has a fixed DTD, which needs to be changed whenever new features are added. XML (and before all of this, SGML) has a dynamic DTD mechanism, in the sense that each document can contain (or refer to) its own DTD. There is a fixed specification for valid DTDs, so a document processor need only “know” how to read DTDs in general when it encounters a new XML (SGML) document with a heretofore unseen DTD. The new document has tags that the processor (e.g., browser) has never seen before (or been told of), but the processor can automatically handle the document because it understands DTDish (my name for the DTD language). DTDish is a language language, and it tells the processor about interpreting the tags in the document. Think of DTDish as a language for instructions. Every (XML/SGML) document comes with a set of instructions pertaining to how the document is to be processed. So long as these instructions are in a language understandable by the processor, it will be possible for the processor to handle the document properly, i.e., according to the instructions.

The reason we don't need a standards organization to define what tags there are and what they mean is that this information is provided anew with

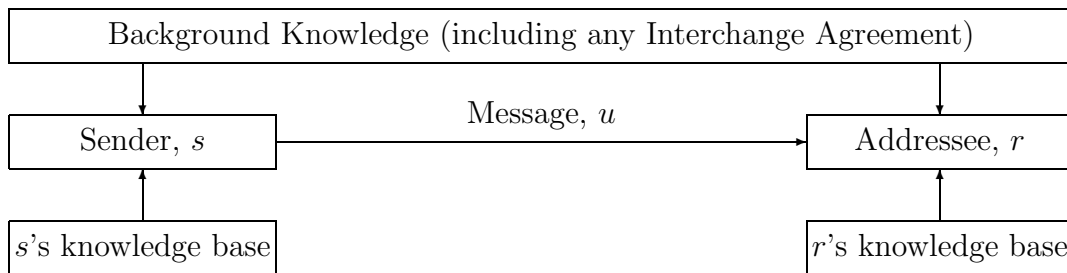


Figure 1: EDI messaging schema: Message u from speaker s to addressee r . From [21].

every document. Anyone can define new tags, or give new definitions to old tags. So long as the definitions are in a valid accompanying DTD, then any processor that understands DTDish will understand the newly defined tags. Thus, XML (and SGML) documents are often said to be “self-describing” [25].

These are lovely ideas. The fact that SGML has been an international standard since 1986, and is widely accepted and used (although not on the Web) attests incontrovertably to its practicality. Nor is there anything importantly special about XML that would make it less useful for electronic commerce than SGML. Quite the contrary. XML is a distilled version of SGML with Web-related additions. XML can be, is being, will be, and probably should be used for EDI purposes in electronic commerce. Even so, it cannot by itself solve the first trade, or transparency, problem in electronic commerce. This, despite widespread intimations and outright claims to the contrary.

To see why XML, nor anything at all like it, cannot solve the transparency problem, we need briefly to consider some fundamentals about communication. With that in hand, I will discuss how, and to what degree, the transparency problem *can* be solved. This in turn will take us back to XML and to an understanding of its proper role in EDI and electronic commerce.

3 Communications Requisites

Consider in the abstract how business messaging—and EDI messaging in particular—is effected. See Figure 1, the basic EDI messaging schema. When

two organizations, s and r , wish to do business with EDI they typically begin by negotiating an Interchange Agreement. This is a contract which, among other things, specifies 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, Background Knowledge (e.g., relevant laws and administrative rules), and the Interchange Agreement to formulate a specific message, u (utterance) in the Figure. Similarly, the receiver of a message draws upon its own knowledge bases and the Background Knowledge (including the Interchange Agreement) in order to interpret and act upon the incoming message, u .

All of this is well and good, and even inevitable. The first trade (or transparency) problem is not a complaint against the basic schema. Rather it is a complaint regarding the cost of constructing the Interchange Agreement, and in particular regarding the cost of fixing the meanings of the messages to be exchanged so that business may be conducted automatically. EDI standards are supposed to reduce this cost. Perhaps they do, but the general consensus is that in fact much manual labor, for analysis and negotiation, is required before two parties will have achieved sufficient agreement to do business by machine. Hence the goal: *design a messaging regime that will support common business transactions and that will allow original messages to be composed by machine, transmitted, and understood by the machine of a new trading partner*. The first trade problem is simply our name for the fact that this goal has not been achieved.

Examining and reflecting on some examples of computer-based communication will help us see certain fundamental principles, which in turn will point towards a solution to the first trade problem. Consider, then, a case of primitive (and very common) computer-to-computer communication. Figure 2, adapted from [26], is an example of a ‘conversation’ between two computers.

1. Computer A: 1, 360
2. Computer B: 6, 350
3. Computer A: 3, 1, 1, 2
4. Computer B: 5, 1, 1
5. Computer A: 3, 1, 1, 3
6. Computer B: 5, 1, 1
7. Computer A: 3, 1, 1, 1
8. Computer B: 4, 1, 1
9. Computer A: 3, 2, 1, 150
10. Computer B: 4, 2, 150
11. Computer A: 6
12. Computer A: 8
13. Computer B: 6

Figure 2: Example of a Process-to-Process Dialog

This is an illustration of what in the trade is called *hand-shaking* between devices, here processes running on computers. But what does it mean? Although the meaning is hardly transparent at first, in fact the exchange is easily decoded.⁵ Here, in Figure 3, is roughly what this means in English.

⁵See [2] for the distinction between decoding and inference.

1. Computer A: Please talk to me on lines 360/361.
2. Computer B: OK. You can talk to me on 350/351.
3. Computer A: Can you do CVSD?
4. Computer B: No, but I can do LPC.
5. Computer A: Can you do RELP?
6. Computer B: No, but I can do LPC.
7. Computer A: How about LPC?
8. Computer B: LPC is fine with me.
9. Computer A: Can you use 150 microsecond sampling?
10. Computer B: I can use 150 microsecond sampling.
11. Computer A: I am ready.
12. Computer A: Are you ready?
13. Computer B: I am ready.

Figure 3: Example of a Process-to-Process Dialog: Translated into English

But how is it that the dialog in Figure 2 comes to have the meaning shown in Figure 3? No mystery at all: In one way or another, the owners of the two computers agreed ahead of time what all the messages would mean, and then they programmed their machines to act appropriately. Points arising:

1. First trade

The communication scheme here could be effected without the two computer owners entering directly into extended discussions regarding what messages there will be and what they will mean. This could happen if someone published a communications protocol adequate for the purposes and it was decided to use that protocol. This would, or could, essentially solve the first trade problem (or its analog in the present case). The solution is rather like what is in fact in place for modem-based communications. Standards are published on how modems are

to do handshaking, and manufacturers simply implement to the standards.

Then why is there a first trade problem in electronic commerce, if there isn't a first handshake problem with Internet service providers?

2. Complex communications

The source of the problem is complexity. In the current example (or range of examples), only a few things need to be said. We arbitrarily name those things, e.g., with numbers, and we publish a table, as it were, of the required names and what they stand for. This is not possible in electronic commerce. There is no way that at any one time we can come usefully close to identifying—to listing in a table—everything that needs to be said. Even if we were to stick to something as basic as a purchase order, there is literally a combinatorial explosion of firms, trade conditions, prices, quantities, items, and so on. Some other approach, other than that evidenced in Figure 2, is required.

Now the second example. Consider a different but still very simple example of computer-based communication. Without loss of generality, the example uses standard Lisp. We begin by defining a symbol—`m` for message—for an arbitrary arithmetic expression:

```
(setq m '(* 5 (- 6 4)))
```

 (1)

If we ask the Lisp interpreter for the value of `m` (by typing the symbol)

```
m
```

 (2)

we see returned its value, the arithmetic expression:

```
(* 5 (- 6 4))
```

 (3)

If we now ask Lisp to evaluate `m`

```
(eval m)
```

 (4)

it correctly calculates the result and returns 10. This utterly normal, commonplace interaction bears reflection in the present context. Remarking on something that is usually quite unremarkable will help us understand the first trade problem. Patience will yield insight.

Notice that in this little interaction we have—or by strong analogy nearly have—created a message, sent the message to a computerized process, and the process has correctly interpreted what the message means. Notice further that so long as the message (the symbol `m`) is well formed, Lisp’s `eval` will interpret it properly, *even if the message expresses an arithmetic formula that happens never to have been created previously during the history of human kind*. Nothing surprising in this, except to note that in a way we have solved the first trade problem. More generally: the message is semantically transparent to the interpreter. Of course, it was built that way. But how does it work? My claim is that a properly general description of what is going on here must also apply to any regime that hopes to solve the transparency (first trade) problem in electronic commerce.

Understood aright, our little Lisp story contains three essential elements for solving our problem. First, the Lisp `eval` function is able to evaluate (interpret) the message because:

1. The message conforms to a formal grammar, which is used by the `eval` function, and
2. Using the grammar the message is composed ultimately of primitive symbols, which `eval` also knows how to evaluate.

In short, there is a *compositional* formal language for the evaluator to work with. Starting with certain basic elements, a grammar is present that specifies how these elements may be composed into larger expressions. Our basic elements can be specified in a lexicon, using BNF. We need the digits, the arithmetic functions, and markers for whitespace.

1. $\langle digit \rangle \longrightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
2. $\langle arith-functor-1 \rangle \longrightarrow + \mid - \mid *$
3. $\langle arith-functor-2 \rangle \longrightarrow /$
4. $\langle S \rangle \longrightarrow \#x20 \mid \#x9 \mid \#xD \mid \#xA \mid \langle S \rangle \langle S \rangle$

The gammar composes messages from the lexicon, using its elements as primitives. Again, we use BNF.

1. $\langle message \rangle \longrightarrow \langle number \rangle \mid (\langle arith-functor-1 \rangle \langle S \rangle \langle message-list \rangle) \mid (\langle arith-functor-2 \rangle \langle S \rangle \langle message-list \rangle \langle S \rangle \langle message-list \rangle)$

2. $\langle message-list \rangle \longrightarrow \langle message \rangle \mid \langle message \rangle \langle S \rangle \langle message-list \rangle$
3. $\langle integer \rangle \longrightarrow \langle digit \rangle \mid \langle digit \rangle \langle integer \rangle$
4. $\langle float \rangle \longrightarrow \langle integer \rangle . \langle integer \rangle$
5. $\langle number \rangle \longrightarrow \langle integer \rangle \mid -\langle integer \rangle \mid \langle float \rangle \mid -\langle float \rangle$

Thus, with a total of 18 primitive symbols in the lexicon, the grammar covers an infinite number of arithmetic expressions. Communicants only need prior agreement on the 18 primitive symbols and the 5 rules in the grammar.

One way or another, any genuinely efficient electronic commerce regime of computer-to-computer communications will require this: a fundamental lexicon plus a formal grammar for composing messages. But we need something else. The second essential facet of this example is broadly called a *semantics*. The lexicon and grammar are necessary for any reasonably rich communication regime, but they are not sufficient. They need to *mean* what we want them to mean. The example to hand works fine for arithmetic expressions, but would not work for other kinds of meanings. If we want to talk about baseball, or move funds between bank accounts, or request payment for services, then in spite of their other nice features, our arithmetic lexicon and grammar just won't do.

So, how do we get a formal language (lexicon + grammar) to mean what we want it to mean? We can stipulate meanings for the terms in the lexicon, just as in our first example. The grammar should be understood as a model, a formal structure whose behavior mimics in relevant ways a certain part of the world. As is true for other kinds of models, we can define a grammar however we will, but then we must engage in empirical investigation to determine whether in fact it maps well to that portion of the world we wish to capture. The matching will typically be at best approximate (think of models in the management sciences). The question of whether the model is sufficiently accurate will typically be resolved only by careful study and judicious decision. In the present example, the decision of whether the language properly models arithmetic expressions is a straightforward one. The larger question, of what a language would have to be to properly model what needs to be said in electronic commerce, is of course much more difficult and not yet resolved. (But I shall have something to say about that in the next sections.)

Finally, there is a third important facet of the current example that is relevant to the discussion of communication for electronic commerce. The grammar more or less completely spans its intended domain, in that every valid arithmetic expression is either recognized by our language, or (if it uses a different syntax) can be automatically transformed into an equivalent expression that is recognized by the language. A language to span all of electronic commerce is a feckless goal. But the other extreme—utterly ad hoc languages, special-built for each application—is a large part of why the first trade problem now looms so large.⁶

With these principles of communication in mind, let us see—if only by glimpse—how we might find our way to a workable solution to the first trade, or transparency, problem in electronic commerce.

4 microFLBC and the Transparency Problem

For some time, I have been engaged in the problem of designing a general-purpose formal language for business communication (FLBC).⁷ To date, a main emphasis of this work has been on the challenge of modeling speech acts, such as promising, requesting, accepting, asserting, and so on. Speech acts are, I believe, central to business communication. A purchase order is a request for payment. An offer is a promise, conditioned on acceptance of the offer. A contract is an exchange of promises, among other things. And so on. Systematic examination of EDI messages reveals that in fact they are closely organized around speech acts [14, 19, 27, 28, 30, 29, 33, 34]. Yet, how to formalize speech acts has been an open question. I have developed the elements of such a theory—based on what I call *lean event semantics*—and have presented it elsewhere (see references above). Recounting the lean event theory is not my aim here. Instead, my aim is to *exploit* the theory for the purpose of making progress on the first trade problem.⁸

Continuing the line of argument from the previous section, I offer here (for the first time⁹) a precisely defined, albeit small, language (lexicon +

⁶See [19] for a detailed discussion of the spanning problem in electronic commerce.

⁷See, e.g., [12, 16, 17, 20, 21] and references cited therein.

⁸Lean event semantics is an extension of what is now called ES Θ theory: event semantics with thematic rôles (aka: subatomic semantics). See [31, 22]; also [14, 13].

⁹Previously, e.g., [20], Moore and I have presented a formal language for business com-

grammar) for electronic commerce. The purpose to hand, recall, is to indicate in a fundamental way how the first trade (or semantic transparency) problem can be satisfactorily resolved. Much work remains to be done and this is not a draft standard for the ISO.

The language to be defined here is microFLBC. It is a fragment of first-order modal logic. I assume a standard formulation of the underlying logic and make no particular assumption regarding which modal system is in play. I have S5 in mind, but the reader is free to pick a favorite system.

Here is the grammar for microFLBC, version 1.0:

1. $\langle \text{microFLBC-utterance} \rangle \longrightarrow (\langle i\text{-force} \rangle \langle S \rangle \wedge \langle S \rangle \square \langle i\text{-content} \rangle)$
2. $\langle i\text{-force} \rangle \longrightarrow (\langle \text{speech-act-predicate} \rangle \langle S \rangle \wedge \langle S \rangle \langle \text{speaker-predicate} \rangle) \mid (\langle i\text{-force} \rangle \langle S \rangle \wedge \langle S \rangle \langle i\text{-force-thematic-role-predicate} \rangle)$
3. $\langle \text{speech-act-predicate} \rangle \longrightarrow \langle \text{speech-act-verb} \rangle (\langle \text{eventuality-ref} \rangle)$
4. $\langle \text{speaker-predicate} \rangle \longrightarrow \text{Speaker} (\langle \text{eventuality-ref} \rangle , \langle S \rangle \langle \text{general-ref} \rangle)$
5. $\langle i\text{-force-thematic-role-predicate} \rangle \longrightarrow \langle i\text{-force-thematic-role} \rangle (\langle \text{eventuality-ref} \rangle , \langle S \rangle \langle \text{general-ref} \rangle) \mid (\langle i\text{-force-thematic-role-predicate} \rangle \langle S \rangle \wedge \langle S \rangle \langle i\text{-force-thematic-role-predicate} \rangle)$
6. $\langle i\text{-content} \rangle \longrightarrow (\langle \text{condition} \rangle \langle S \rangle \rightarrow \langle S \rangle (\langle \text{speech-act-auxiliary-predicate} \rangle \langle S \rangle \leftrightarrow \langle S \rangle \langle \text{content} \rangle))$
7. $\langle \text{condition} \rangle \longrightarrow \top \mid \langle \text{speech-act-auxiliary-predicate} \rangle$
8. $\langle \text{speech-act-auxiliary-predicate} \rangle \longrightarrow \langle \text{speech-act-auxiliary} \rangle (\langle \text{eventuality-ref} \rangle)$
9. $\langle \text{content} \rangle \longrightarrow \langle \text{microFLBC-utterance} \rangle \mid \langle \text{simple-content} \rangle \mid (\langle \text{content} \rangle \langle S \rangle \wedge \langle S \rangle \langle \text{content} \rangle)$
10. $\langle \text{simple-content} \rangle \longrightarrow (\langle \text{simple-content-verb-predicate} \rangle) \mid (\langle \text{simple-content-verb-predicate} \rangle \langle S \rangle \wedge \langle S \rangle \langle \text{content-predicates} \rangle)$

munication. That language, however, is what I now call an *application FLBC*. Here, the language being defined aims at capturing the fundamental semantics for the application, and I call it a *semantic FLBC*. The relationship between the two will be clarified in the following section when we return to XML.

11. $\langle \text{simple-content-verb-predicate} \rangle \longrightarrow \langle \text{ordinary-verb-predicate} \rangle \mid \langle \text{speech-act-auxiliary-predicate} \rangle$
12. $\langle \text{content-predicates} \rangle \longrightarrow \langle \text{content-predicte} \rangle \mid (\langle \text{content-predicte} \rangle \langle S \rangle \wedge \langle S \rangle \langle \text{content-predicte} \rangle)$
13. $\langle \text{content-predicte} \rangle \longrightarrow \langle \text{thematic-role-predicate} \rangle \mid \langle \text{prepositional-predicate} \rangle \mid \langle \text{misc-predicate} \rangle \mid \langle \text{domain-specific-predicate} \rangle$
14. $\langle \text{ordinary-verb-predicate} \rangle \longrightarrow \langle \text{ordinary-verb} \rangle \langle \text{eventuality-ref} \rangle$
15. $\langle \text{thematic-role-predicate} \rangle \longrightarrow \langle \text{thematic-role} \rangle \langle \text{event-arg-pair} \rangle$
16. $\langle \text{event-arg-pair} \rangle \longrightarrow (\langle \text{eventuality-ref} \rangle , \langle S \rangle \langle \text{general-ref} \rangle)$
17. $\langle \text{prepositional-predicate} \rangle \longrightarrow \langle \text{preposition} \rangle \langle \text{event-arg-pair} \rangle$
18. $\langle \text{misc-predicate} \rangle \longrightarrow \langle \text{misc1} \rangle \langle \text{general-ref} \rangle \mid \langle \text{misc2} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle) \mid \langle \text{misc3} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle)$
19. $\langle \text{domain-specific-predicate} \rangle \longrightarrow \langle \text{domain-specific1} \rangle \langle \text{general-ref} \rangle \mid \langle \text{domain-specific2} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle) \mid \langle \text{domain-specific3} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle)$
20. $\langle \text{general-ref} \rangle \longrightarrow \text{PCDATA} \mid \langle \text{function1} \rangle \langle \text{general-ref} \rangle \mid \langle \text{function2} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle) \mid \langle \text{function3} \rangle (\langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle , \langle \text{general-ref} \rangle)$
21. $\langle \text{eventuality-ref} \rangle \longrightarrow \langle \text{general-ref} \rangle$

Lexicon 1.0 for microFLBC. Terminals/lexicon for microFLBC:

1. $\langle S \rangle \longrightarrow \#x20 \mid \#x9 \mid \#xD \mid \#xA \mid \langle S \rangle \langle S \rangle$
2. $\langle \text{speech-act-verb} \rangle \longrightarrow \text{promise} \mid \text{assert} \mid \text{declare} \mid \text{request} \mid \text{purchase-order} \mid \text{cancel} \mid \text{confirm} \mid \text{endorse} \mid \text{invoice} \mid \text{inquire} \mid \text{estimate} \mid \text{order} \mid \text{report} \mid \text{specify} \mid \text{clear} \mid \text{approve} \mid \text{void} \mid \text{discharge}$
3. $\langle \text{ordinary-verb} \rangle \longrightarrow \text{pay} \mid \text{deliver} \mid \text{ship} \mid \text{arrive} \mid \text{become-due} \mid \text{become-effective} \mid \text{change} \mid \text{examine} \mid \text{count} \mid \text{create} \mid \text{expire} \mid \text{fail} \mid \text{manufacture} \mid \text{open} \mid \text{close} \mid \text{perform} \mid \text{place} \mid \text{process} \mid \text{arrive} \mid \text{depart} \mid \text{load} \mid \text{unload} \mid \text{receive}$

4. $\langle i\text{-force-thematic-role} \rangle \longrightarrow \textit{Addressee} \mid \textit{Theme} \mid \textit{Cul} \mid \textit{Comc} \mid \textit{Hold} \mid \textit{Sake}$
5. $\langle \textit{speech-act-auxiliary} \rangle \longrightarrow V \mid K \mid H \mid \textit{Auth}$
6. $\langle \textit{thematic-role} \rangle \longrightarrow \textit{Agent} \mid \textit{Performer} \mid \textit{Experiencer} \mid \textit{Benefactive} \mid \textit{Goal} \mid \textit{Theme} \mid \textit{Sake} \mid \textit{Cul} \mid \textit{Comc} \mid \textit{Hold} \mid \textit{Location} \mid \textit{Source} \mid \textit{Instrument}$
7. $\langle \textit{preposition} \rangle \longrightarrow \textit{to} \mid \textit{from} \mid \textit{into} \mid \textit{at} \mid \textit{for} \mid \textit{with} \mid \textit{by} \mid \textit{in}$
8. $\langle \textit{misc1} \rangle \longrightarrow \textit{red} \mid \textit{green} \mid \textit{blue}$
9. $\langle \textit{misc2} \rangle \longrightarrow \leq \mid = \mid < \mid > \mid \geq \mid \textit{description} \mid \textit{during} \mid$
10. $\langle \textit{misc3} \rangle \longrightarrow \textit{unit} \mid \textit{quantity} \mid \textit{unitprice}$
11. $\langle \textit{function1} \rangle \longrightarrow \textit{day}$
12. $\langle \textit{function2} \rangle \longrightarrow + \mid - \mid * \mid /$
13. $\langle \textit{domain-specific2} \rangle \longrightarrow \textit{FOB} \mid \textit{TERMS}$

4.1 Representing a simple promise

Our first example is a simple promise, of the sort that is made many times daily in the conduct of commerce.

At time t , s promises r that s will deliver to r goods g within 30 days.

(5)

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

1. `promise : 12345`
2. `date-time : 1999-09-23`
3. `from : s`
4. `to : r`

5. deliver

(a) goods : g

(b) to : r

(c) by : s

(d) date-time : 1999-09-23 + day(1999-09-23 + 30)

(6)

So, we should think of expression (6) as an instance of u in Figure 1.

Using my lean event semantics [16] and microFLBC, expression (5) is rendered into first-order modal logic as follows:

$$\begin{aligned} & \text{promise}(12345) \wedge \text{Speaker}(12345, s) \wedge \text{Addressee}(12345, r) \wedge \\ & \text{Cul}(12345, 1999-09-23) \wedge \Box(\top \rightarrow (K(12345) \leftrightarrow (\text{deliver}(e) \wedge \\ & \text{Agent}(e, s) \wedge \text{Benefactive}(e, r) \wedge \text{Sake}(e, 12345) \wedge \text{Theme}(e, g) \\ & \wedge \text{Cul}(e, t) \wedge \leq(t, +(1999-09-23, \text{day}(1999-09-23 + 30)))))) \end{aligned}$$

(7)

Comments and points arising:

1. Expression (7) reads roughly as “12345 is a promise by s to r , occurring on September 23, 1999. The promise 12345 concerns a delivery event e . Necessarily, this promise is kept if and only 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 thirty days of September 23, 1999.”
2. This reading is also (I claim) a correct interpretation of the EDI-like message, Expression (6) and of the original ordinary language-like utterance, Expression (5).
3. \Box in Expression 7 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 I elide. It is generally safe simply to drop the operator, \Box . Doing this yields the *extensional approximation*, and for practical purposes the approximation suffices. Sticking to principles, the grammar for microFLBC retains the requirement that the \Box be present.

4. Unlike Expression (6), Expression (7) is logical and fully formal. Note in this regard how Expression (7) makes explicit so much that would otherwise have to be read into or assumed regarding Expressions (5) and (6), which are open to more than one interpretation.
5. The predicates in Expression (7) all occur in the lexicon, and are of general utility. They could be used in representing 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 [12, 13, 14, 15, 16].
7. Two kinds of semantic theory are present: a theory applying to natural language and a theory applying to first-order modal logic. Expression (7) exploits lean event semantics, a partial theory of natural language semantics, to represent a meaning in logic. Lean event semantics extends $ES\Theta$ theory [31, 22]. See also [14, 13]. I am assuming (and have argued elsewhere) that this theory provides us with an adequate account of the semantics of the sorts of expressions exhibited for discussion here (i.e., messages in electronic commerce). If it does, then—I am now arguing—we can see a way through to solving the first trade problem in electronic commerce. The second semantic theory to hand is the semantics for first-order modal logic. This theory is well-established and I take it to be unproblematic.
8. \top , appearing in item (7) of the microFLBC grammar, is shorthand for any logical tautology. It always evaluates to true. Anything conditioned on \top is thus only vacuously conditioned. If I say I will go to the store whether or not it rains, my going is but vacuously conditioned on the weather.
9. PCDATA, appearing in item (20) of the microFLBC grammar, alludes to the XML use of that expression: basically PCDATA matches any string of characters. This is where we give up specifying structure and content ourselves with names of things.

10. microFLBC contains very little by way of validation principles. Data types are not recognized and no real distinction is made between names of eventualities (events, processes, and states) and names of other kinds of things (e.g., people and products). Every utterance act must have a *Speaker*, but little else is required. These are important matters indeed, but they are beyond the scope of this paper.

Now to a more complex example.

TO: r		CUSTOMER NO. s			
		TERMS 30-days-net			
SHIP TO: an-address		SALES			
		SHIP WEEK OF 1999-12-03			
		FOB customer			
ORDER NO. 54321	DELIVERY VIA		ROUTING		
PLEASE SHIP THE FOLLOWING AS SPECIFIED					
ITEM	QUANTITY ORDERED	DESCRIPTION	UNIT COUNT	UNIT PRICE	TOTAL AMOUNT
catid-32-9	box	copier paper	6	\$45.01	\$270.06
catid-35-9	dozen	no. 2 pencil	3	\$1.99	\$5.97
SPECIAL INSTRUCTIONS:		TOTAL AMOUNT			\$276.03
		DATE	APPROVAL SIGNATURE		
		1999-11-19			
		PURCHASER SIGNATURE			
		TITLE			
*** PURCHASE ORDER ***					

Figure 4: Simple Purchase Order Example

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
 - (c) description: "copier paper"
 - (d) quantity: 6
 - (e) unit-price: 45.01
 - (f) line-total: 270.06
11. deliver
 - (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 5: EDI-like representation of the simple purchase order in Figure 4.

4.2 Representation of a Simple Purchase Order

Figures 4 and 5 show, respectively, a simple example of a specific purchase order and a representation of it in an EDI-like (structured, not logical) syntax. I have kept the names of data items short, and the number of data items few, for the sake of convenience. The example is rich enough to bear the points I wish to make, and sparse enough to avoid burdening 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 6, page 22, displays the results of using lean event semantics and microFLBC to represent the purchase order of Figures 4 and 5. Although there is very much to say about this example, I shall give just a few remarks, aimed at making the example accessible and its connection with the main points clear.

1. The expression in Figure 6 is a fully formal, logical *model* (and hence, approximation) of what I think would typically be meant by the purchase order shown in Figure 4.
2. As in the previous example:
 - (a) Predicates beginning with a capital letter (*Speaker*, *Theme*, *Comc*, etc.) are generic reserved words in lean event semantics;
 - (b) Predicates in lower case (e.g., *deliver*, *pay*) are (here) verbs whose meaning is specified independently (more on this below); and
 - (c) Predicates in all upper case (e.g., *FOB*, *TERMS*) are domain-specific.

All of these predicates appear in the microFLBC lexicon.

3. The expression in Figure 6 is a single logical conjunction with four main blocks of code.
4. The gist of the first block is that this is a purchase order from *s* to *r* happening (“culminating,” *Cul*) on 1999-11-19.
5. The gist of the second and third blocks is to indicate 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

$\text{purchase-order}(54321) \wedge \text{Speaker}(54321, s) \wedge \text{Addressee}(54321, r) \wedge$
 $\text{Cul}(54321, 1999-11-19) \wedge$

$\square((\top \rightarrow (H(54321) \leftrightarrow$

$((\text{ship}(e1) \wedge \text{Agent}(e1, r) \wedge \text{Benefactive}(e1, s) \wedge \text{to}(e1, \text{an-}$
 $\text{address}) \wedge \text{Theme}(e1, \text{catid-32-9}) \wedge \text{Sake}(e1, 54321) \wedge \text{unit}(e1,$
 $\text{catid-32-9, box}) \wedge \text{description}(\text{catid-32-9, "copier paper"}) \wedge$
 $\text{quantity}(e1, \text{catid-32-9, 6}) \wedge \text{Comc}(e1, t1) \wedge \text{FOB}(e1, \text{customer})$
 $\wedge \text{during}(t1, \text{week-of}(1999-12-03))) \wedge$

$(\text{ship}(e2) \wedge \text{Agent}(e2, r) \wedge \text{Benefactive}(e2, s) \wedge \text{to}(e2, \text{an-}$
 $\text{address}) \wedge \text{Theme}(e2, \text{catid-35-9}) \wedge \text{Sake}(e2, 54321) \wedge \text{unit}(e2,$
 $\text{catid-35-9, dozen}) \wedge \text{description}(\text{catid-35-9, "no. 2 pencil"}) \wedge$
 $\text{quantity}(e2, \text{catid-35-9, 3}) \wedge \text{Comc}(e2, t2) \wedge \text{FOB}(e2, \text{customer})$
 $\wedge \text{during}(t2, \text{week-of}(1999-12-03))) \wedge$

$(H(54321) \rightarrow (K(54321) \leftrightarrow$

$(\text{pay}(e3) \wedge \text{Agent}(e3, s) \wedge \text{Benefactive}(e3, r) \wedge \text{Sake}(e3, [e1,$
 $e2]) \wedge \text{Theme}(e3, p3) \wedge \text{unit}(e3, p3, \$) \wedge \text{quantity}(e3, p3, q)$
 $\wedge \text{unitprice}(e1, \text{catid-32-9, p1}) \wedge \text{unitprice}(e2, \text{catid-35-9, p2}) \wedge$
 $\text{unit}(e1, p1, \$) \wedge \text{unit}(e2, p2, \$) \wedge \text{quantity}(e1, p1, 45.01) \wedge$
 $\text{quantity}(e2, p3, 1.99) \wedge =(p3, +(*(45.01, 6), *(1.99, 3))) \wedge$
 $\text{Cul}(e3, t3) \wedge \text{TERMS}(e3, 30\text{-days-net}))))$

Figure 6: Representation, via lean event semantics, of a simple purchase order, Figures 4 and 5

of these events follow and match to the original purchase order. Note that in both cases, the shipping events are (requested to) begin (“commencing”, *Comc*) during the period indicated. Nothing is said about when delivery should occur. Also, both both events are vacuously conditioned on \top , a generic symbol for something that is always true. Vacuous conditioning is no conditioning at all.

6. The gist of the fourth block is that if the afore-described request is in fact honored (here the conditioning is nonvacuous), then this purchase order is also making a promise. The promise is kept (K) if and only if $e\beta$ is a paying event with the properties described.
7. Note that the 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 pay r as described. It follows logically (as it should) that the promise associated with the purchase order, made by s , has not been kept, i.e., that $\neg K(54321)$.

4.3 Mapping to the world

No semantics can ever be entirely formal. At some point we need 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 rely as much as possible on composition of symbols to provide additional meaning. With our logical semantics (e.g., Figure 6) we need to provide a basic mapping for: terms (s , r , etc.), functions (e.g., *week-of*), and predicates (e.g., *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, 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 one defines them by composition from more primitive functions.

Predicates, as noted, are of three kinds:

1. Specific for lean event semantics, e.g., *Theme*, *Comc*, *Unit*
2. Generic, e.g., *ship*, *pay*
3. Domain specific, e.g., *FOB*, *TERMS*

The specific predicates should be few in number and can be defined (mapped) explicitly. The generic predicates can and should be mapped to a clear, broadly-accepted, and public specification. Language having the importance it does, such specifications are indeed available. WordNet, an electronic lexicon produced over a number of years at Princeton University, is an excellent example [8]. Let us see, briefly, what WordNet has to say about the three generic (verb) predicates in Figure 6: *purchase-order*, *ship*, *pay*.

WordNet recognizes *purchase order* as a noun, but not as 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} <noun.communication> order1#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")
```

Notice that a purchase order here—and in Figure 6—involves both a request for something and a (promised) payment for it.

WordNet 1.6 recognizes *ship* as a verb for which there is exactly one sense: to transport commercially.

1 sense of ship

Sense 1

```
{01328437} <verb.motion> transport1#4, send#4, ship#1 --
(transport commercially)
=> {01328337} <verb.motion> barge1#2 --
    (transport by barge on a body of water)
=> {01331285} <verb.motion> dispatch#1, despatch#1,
    send off#1 -- (send off promptly)
=> {01331167} <verb.motion> bundle off#1 --
    (send off unceremoniously)
=> {01331450} <verb.motion> route2#1 --
    (send documents or materials to appropriate
    destinations)
=> {01331576} <verb.motion> forward#1, send on#1 --
    (send or ship onward from an intermediate post or
    station in transit; "forward my mail")
```

More refined meanings are available with different verbs. We may thus tentatively identify the *ship* of Figure 6 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> give1#5, pay#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 up#3,
compensate2#4 -- (do or give something to somebody in
return; "Does she pay you for the work you are doing?")
5. {01695538} <verb.social> pay#5 --
(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> yield#10, pay1#7, bear1#8 --
(bring in; as of investments; "interest-bearing accounts";
"How much does this savings certificate pay annually?")
8. {01869530} <verb.stative> pay#8 --
(be worth it; "It pays to go through the trouble")
9. {00496485} <verb.cognition> give2#10, pay#9, devote#2 --
(as in the expressions "give thought to";
"give priority to", etc.)
10. {01541783} <verb.possession> pay14#10 --
(discharge or settle; "pay a debt"; "pay an obligation")
11. {01600647} <verb.possession> pay13#11 --
(make a compensation for; "a favor that cannot
be paid back")

Pretty straightforwardly, we can tentatively identify microFLBC's *pay* with

WordNet’s *pay#1*.

If our lexicon is to be properly completed, this sort of mapping must be carried through systematically. Further excursions in that direction, however, would only divert us from our main subject.

5 Back to XML

It will help to summarize before returning to discussion of XML. Through a series of examples I have observed that effective computer-to-computer (automated) communication requires:

1. A *common* lexicon, which identifies the primitive expressions and what they mean

This is what makes possible the simple sort of communication illustrated, above, in Figure 2. If we are content with such an elementary form of communication, then the transparency problem can (in principle) be solved simply by making available to all parties the common lexicon. Each may then program its own computers without consulting the other, and (in principle) the first exchange and all subsequent exchanges will work properly.

The problem is that for most purposes, especially for electronic commerce, a “words only”—or *atomic*—form of exchange is insufficiently rich, and we must resort to messages *composed* from items in the lexicon.

2. A *common* grammar that specifies the permitted compositional expressions

Lacking this, it would border on miraculous if the sender composed messages that the receiver could recognize correctly.

3. A *common* convention for interpreting what the expressions produced by the common grammar mean

This is also known as a semantics for the (common) language. In the two relevant examples above, the Lisp arithmetic expressions and microFLBC, I alluded to but did not specify the associated semantics. For the case of the Lisp expressions, it is the semantics that tells us that applying the functor $+$ to a list of numbers produces their sum,

instead of something else. microFLBC, being a fragment of first-order modal logic, inherits the semantics for that language. This is what tells us that \wedge means (roughly, *and* but a truth table can be given to make this rigorous), and indeed what each possible expression in the grammar (BNF) means.¹⁰

With these points to hand, we can diagnose the sources of the transparency (spontaneity, or first-trade) problem in EDI as:

1. Lack of a clear and complete lexicon

Nothing resembling even the above mapping of a few predicates to WordNet is generally available, although in certain domains much has been achieved in this direction.

2. No fully-specified grammar

The various EDI standards (X12, EDIFACT, etc.) attempt to define compositional expressions, and hence can be seen as providing grammars for business communication. Two major shortcomings are present. First, the grammars are incomplete; they do not fully and rigorously specify the permitted messages. This is often an important locus of negotiation for organizations preparing for their first electronic trade. Second, grammars have proliferated. The standards each identify scores of message types (invoice, purchase order, request for quotation, etc.), called *transaction sets*. Individual ‘grammars’ (speaking loosely) are defined for each transaction set. This, too, only increases the pre-trade clarification burden.

3. No semantics

Or nearly no semantics. The standards provide rough and intuitive—and occasionally quite specific—information on what the various fields are for and what can go in them, but this is a very long way from having a full and rigorous semantics. Even if the trading parties are determined to follow the standards exactly, this is impossible because the standards are not exact. Negotiation ensues.

¹⁰Regarding *common*, I oversimplify for the sake of the issues at hand. I do not want to suggest that communicants must be looking at exacting the same lexicon, etc. They could have in common different copies of things. Also, they might come to acquire their copies by incremental learning. Skeyrms [35, chapter 5] is particularly convincing on this subject, but we are far from seeing our way to applications via learning of meaning.

What of XML? It indeed brings much to our table. First, the DTD mechanism offers the prospect of a fully rigorous grammar for business messaging. What a DTD really is is an executable BNF specification. At least in principle (and surely for very many practical purposes), this answers well to the requirement for a formal grammar. In this regard alone, XML presents the prospect of a great improvement over current EDI specification regimes. Second, XML has a namespace mechanism by which it can fix the references of the terms it uses. For example, a catalog number for a product need not be universally unique. If it is locally unique—within a given catalog—and we have a way to refer to the catalog in question, then the product number can (in principle) be used without ambiguity. Exploring this aspect of XML in detail is beyond the scope of this paper, but it is safe to say that the namespace mechanism is at least a promising vehicle for solving the lexicon requirement.

This leaves the question of semantics. Just as a BNF provides the grammar or syntax for a language, and cannot provide the semantics, so XML's DTD mechanism cannot provide a semantics either. As with namespaces, an XML document can point to a semantic specification, but it cannot provide one for itself. What to do? We can give an XML document a semantics by mapping its DTD to another language which itself is or has a formal semantics. What is special (although not unique) about logic is that it can serve as such a language. Semantics for logic is well worked out, and arguably the microFLBC fragment can properly express much of what we need to say for business messaging. So, the trick is to define XML DTDs that can be mapped to microFLBC. Then, microFLBC (or something like it) and the mapping correspondence with the DTD can together serve as the common semantics. This in principle completes our three requirements for effective communication and—in principle—lets us solve the transparency problem.

I will illustrate with a simple example. Suppose s wants to send a message to a requesting payment for items delivered in response to a previous work order, number 789, originally issued by a . In English the message is:

s requests of a that a pay s for item 789.

In microFLBC this is more carefully stated as

$$\begin{aligned} & \text{request}(2345) \wedge \text{Speaker}(2345, s) \wedge \text{Addressee}(2345, a) \wedge \Box (\top \\ & \rightarrow (H(2345) \leftrightarrow (\text{pay}(23456) \wedge \text{Agent}(23456, a) \wedge \text{Benefactive}(23456, \\ & s) \wedge \text{Sake}(23456, 789))) \end{aligned}$$

Here is one way to say this in XML:

```
<microFLBC-utterance>
  <i-force>
    <speech-act-predicate>
      <speech-act-verb>request</speech-act-verb>
      <eventuality-ref>2345</eventuality-ref>
    </speech-act-predicate>
    <speaker-predicate>
      <eventuality-ref>2345</eventuality-ref>
      <general-ref>s</general-ref>
    </speaker-predicate>
    <i-force-thematic-role-predicate>
      <i-force-thematic-role>Addressee</i-force-thematic-role>
      <eventuality-ref>2345</eventuality-ref>
      <general-ref>r</general-ref>
    </i-force-thematic-role-predicate>
  </i-force>
  <i-content>
    <condition>True</condition>
    <speech-act-auxiliary-predicate>
      <speech-act-auxiliary>H</speech-act-auxiliary>
      <eventuality-ref>2345</eventuality-ref>
    </speech-act-auxiliary-predicate>
    <content>
      <simple-content>
        <simple-content-verb-predicate>
          <ordinary-verb-predicate>pay</ordinary-verb-predicate>
          <eventuality-ref>23456</eventuality-ref>
        </simple-content-verb-predicate>
      </simple-content>
    </content>
  </i-content>
</microFLBC-utterance>
```

```

</simple-content-verb-predicate>
<content-predicates>
  <content-predicate>
    <thematic-role-predicate>
      <thematic-role>Agent</thematic-role>
      <event-arg-pair>
        <eventuality-ref>23456</eventuality-ref>
        <general-ref>a</general-ref>
      </event-arg-pair>
    </thematic-role-predicate>
  </content-predicate>
  <content-predicate>
    <thematic-role-predicate>
      <thematic-role>Benefactive</thematic-role>
      <event-arg-pair>
        <eventuality-ref>23456</eventuality-ref>
        <general-ref>s</general-ref>
      </event-arg-pair>
    </thematic-role-predicate>
  </content-predicate>
  <content-predicate>
    <thematic-role-predicate>
      <thematic-role>Sake</thematic-role>
      <event-arg-pair>
        <eventuality-ref>23456</eventuality-ref>
        <general-ref>789</general-ref>
      </event-arg-pair>
    </thematic-role-predicate>

```

```
        </content-predicate>
    </content-predicates>
</simple-content>
</content>
</i-content>
</microFLBC-utterance>
```

There is a more or less transparent mapping between this XML expression and microFLBC. I leave it to the reader as an exercise.

6 Conclusion

If communicating automated agents are to benefit from drastically reduced first-trade costs, it will be necessary to provide them (and their human masters) with:

1. a fundamental semantic language for modeling communication, including
 - (a) a lexicon
 - (b) a grammar
 - (c) a semantics
2. an application language, for business communication,
3. a mapping between the two languages

and all of this must be fully formal and processable by machine. This is a tall order and very much remains to be done. Fortunately, what we have but glimpsed here can be realized in increments. How much would microFLBC have to be expanded in order to serve as the semantic foundation for some domain of application, heretofore poorly served by EDI? The prospects, I think, are quite good, but that is something for another paper.

References

- [1] Nabil R. Adam and Yelena Yesha, editors. *Electronic Commerce: Current Research Issues and Applications*, volume 1028 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 1996. ISBN: 3-540-60738-2.
- [2] Hemant K. Bhargava and Steven O. Kimbrough. On embedded languages, meta-level reasoning and computer-aided modeling. In Stephen G. Nash and Ariela Sofer, editors, *The Impact of Emerging Technologies on Computer Science and Operations Research*, pages 27–44. Kluwer Academic Publishers, Boston, MA, 1995. ISBN 0-7923-9542-5. File: csts-94-meta-sok-hkb.
- [3] Roger W.H. Bons, Ronald M. Lee, and Rene W. Wagenaar. Computer-aided auditing of inter-organizational trade procedures. *Intelligent Systems in Accounting, Finance and Management*, 1997.
- [4] Kevin Dick. *XML: A Manager's Guide*. Addison-Wesley, Reading, Massachusetts, 2000. ISBN: 0-201-43335-4.
- [5] Chris Driscoll. Xml touted as cure for edi ills: New markup language extends web capabilities beyond HTML's limits. Web page, 5 August 1997. <http://www.geocities.com/WallStreet/Floor/5815/edinews01.htm>, accessed 1999-12-28.
- [6] Margaret A. Emmelhainz. *EDI: A Total Management Guide*. Van Nostrand Reinhold, New York, NY, second edition, 1993. ISBN: 0-442-312690-9.
- [7] Margaret A. Emmelhainz. Electronic data interchange in logistics. In James F. Robeson and William C. Copacino, editors, *The Logistics Handbook*, pages 737–756. The Free Press, New York, NY, 1994. ISBN: 0-02-926595-9.
- [8] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA, 1998. ISBN: 0-262-06197-X.
- [9] Charles F. Goldfarb and Paul Prescod. *The XML Handbook*. Prentice Hall PTR, Upper Saddle River, NJ, 1998. ISBN: 0-13-081152-1.

- [10] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francisco, CA, 1998. ISBN: 1-55860-495-2.
- [11] Paul Kimberley. *Electronic Data Interchange*. McGraw-Hill, Inc., New York, New York, 1991.
- [12] Steven O. Kimbrough. On representation schemes for promising electronically. *Decision Support Systems*, 6(2):99–122, 1990.
- [13] Steven O. Kimbrough. On electronic commerce, subatomic semantics and Cæsar’s stabbing. In Ralph H. Sprague, Jr., editor, *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, Los Alamitos. CA, 1997. IEEE Press. File: hicss97-sok-subatomic-final.
- [14] Steven O. Kimbrough. On $ES\Theta$ theory and the logic of the X12 date/time qualifiers. In Ralph H. Sprague, Jr., editor, *Proceedings of the Thirty-First Hawai’i International Conference on System Sciences*, Los Alamitos. CA, 1998. IEEE Press. File: flbcx12.tex.
- [15] Steven O. Kimbrough. Sketch of a basic theory for a formal language for business communication. In Ralph H. Sprague, Jr., editor, *Proceedings of the Thirty-First Hawai’i International Conference on System Sciences*, Los Alamitos. CA, 1998. IEEE Press. File: flbch98.tex.
- [16] Steven O. Kimbrough. Formal language for business communication: Sketch of a basic theory. *International Journal of Electronic Commerce*, 3(2):23–44, Winter 1998–99.
- [17] Steven O. Kimbrough and Ronald M. Lee. On illocutionary logic as a telecommunications language. In *Proceedings of the International Conference on Information Systems*, pages 15–25, 1986.
- [18] Steven O. Kimbrough and Scott A. Moore. On obligation, time, and defeasibility in systems for electronic commerce. In Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., editors, *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences, Volume III, Information Systems: DSS/Knowledge-Based Systems*, pages 493–502, Los Alamitos, California, 1993. IEEE Computer Society Press.
- [19] Steven O. Kimbrough and Scott A. Moore. On the spanning hypothesis for EDI semantics. In Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr.,

editors, *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*, Los Alamitos, California, January 1993. IEEE Computer Society Press.

- [20] Steven O. Kimbrough and Scott A. Moore. On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity. *ACM Transactions on Information Systems*, 15(4):321–367, October 1997.
- [21] Steven O. Kimbrough and Yao-Hua Tan. On lean messaging with unfolding and unwrapping for electronic commerce. *International Journal of Electronic Commerce*, forthcoming 2000.
- [22] Richard Larson and Gabriel Segal. *Knowledge of Meaning: An Introduction to Semantic Theory*. The MIT Press, Cambridge, Massachusetts, 1995. ISBN: 0-262-62100-2.
- [23] Ronald M. Lee. Distributed electronic trade scenarios: Representation, design, prototyping. *International Journal of Electronic Commerce*, 3(2), 1999.
- [24] Fritz Lehmann. Machine-negotiated, ontology-based EDI (electronic data interchange). In Nabil R. Adam and Yelena Yesha, editors, *Electronic Commerce: Current Research Issues and Applications*, volume 1028 of *Lecture Notes in Computer Science*, pages 27–45. Springer, Berlin, Germany, 1996. ISBN: 3-540-60738-2.
- [25] Richard Light. *Presenting XML*. SAMS Net, Indianapolis, IN, 1997. ISBN: 1-57521-334-6.
- [26] David McDonald. Conversations between programs. In Lucia Vaina and Jaakko Hintikka, editors, *Cognitive Constraints on Communication*, volume 18 of *Synthese Language Library*, pages 403–424. D. Reidel Publishing Company, Boston, MA, 1985. ISBN: 90-277-1456-8.
- [27] Scott A. Moore. *Saying and Doing: Uses of Formal Languages in the Conduct of Business*. PhD thesis, University of Pennsylvania, The Wharton School, Department of Operations and Information Management, 3620 Locust Walk, Suite 1300, Philadelphia, PA 19104-6366, December 1993.

- [28] Scott A. Moore. Categorizing automated messages. *Decision Support Systems*, 22(3):213–241, 1998.
- [29] Scott A. Moore. A foundation for flexible automated electronic commerce. *Information Systems Research*, forthcoming.
- [30] Scott A. Moore. KQML & FLBC: contrasting agent communication languages. *International Journal of Electronic Commerce*, forthcoming 2000.
- [31] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. Current Studies in Linguistics. The MIT Press, Cambridge, MA, 1990. ISBN: 0-262-66093-8.
- [32] Airi Salminen. EDIFACT for business computers: Has it succeeded? *StandardView*, 3(1):33–42, March 1995.
- [33] Munidar P. Singh. A semantics for speech acts. *Annals of Mathematics and Artificial Intelligence*, 8(I–II):47–71, 1993. Reprinted in [10].
- [34] Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, December 1998.
- [35] Brian Skyrms. *Evolution of the Social Contract*. Cambridge University Press, New York, NY, 1996. ISBN: 0-521-55583-3.
- [36] K. Steel. Another approach to standardising EDI. *Electronic Markets*, 12, 1994.
- [37] Ken Steel. The standardisation of flexible edi messages. In Nabil R. Adam and Yelena Yesha, editors, *Electronic Commerce: Current Research Issues and Challenges*, pages 13–26. Springer-Verlag, Berlin, Germany, 1996. ISBN 3-540-60738-2.
- [38] Yao-Hua Tan and Walter Thoen. A logical model of transfer of obligations in trade contracts. *Journal of Accounting, Management and Information Systems*, forthcoming 1999.
- [39] Yao-Hua Tan and Walter Thoen. Modeling directed obligations and permissions in trade contracts. In Jay F. Nunamaker, Jr. and Ralph H. Sprague, Jr., editors, *Proceedings of the Thirty-First Annual Hawai'i International Conference on System Sciences*. IEEE Press, January 1998.

[40] The XML/EDI Group. Home page for the XML/EDI Group. Web page, December 1999. <http://www.xmledi.net>.