Introducing the Fair and Logical Trade Project

Steven O. Kimbrough
and Alan S. Abrahams
University of Pennsylvania
Philadelphia, USA
{sok,asa28}@wharton.upenn.edu

Andrew J.I. Jones
King's College London
London, UK
ajijones@dcs.kcl.ac.uk

David M. Eyers
and Jean M. Bacon
University of Cambridge
Cambridge, UK
{dme26,jmb25}@cl.cam.ac.uk

Abstract

We introduce our framework for logic-based compositional e-commerce interaction. We aim to provide open-source software which adds a light-weight formal messaging layer to business communications, to increase the accessibility of e-commerce infrastructure to smaller business players. In the process we hope to develop a comprehensive theory of business communication. We present the logical structures and techniques we apply, and provide initial prototype testing results.

1 Introduction

The Fair and Logical Trade (F&LT) project is recently formed. At present it involves a number of researchers at several different universities. We aim to make scientific advances and to produce usable, and used, open source software, and conceive of the project as an exercise in *embodied research*, research that produces, in part, usable objects, whose actual use provides testing and validation of the research itself. (See [17] for a brief discussion of the concepts of embodied research.) As such, we are open to, and invite, broader participation. The purpose of this position statement is to do just that. In what follows, we briefly overview the broader aims and context of the project. Following that, and for most of the paper, we provide a sketch of certain aspects of our technical thinking.

The F&LT project has two broadly conceived goals. The first is to design, implement, and field light-weight business application software for inter-organizational (and preferably, international) transactions. Prototypical transactions for the project would be purchase orders, invoices, bills of lading, and other essential types of messages for conducting commerce. These types of messages are presently addressed by EDI (electronic data interchange) systems and standards, and their more recent descendants, e.g., ebXML. (cf., `http://www.ebxml.org/`, `http://xml.coverpages.org/ebXML.html`, `http://www.ebxmlforum.org/`). Also prototypically, the users of this software would be the sorts of buyers and sellers who today participate in "fair trade" international commerce. For examples, see: Fairtrade (`http://www.fairtrade.org.uk/`, also `http://www.fairtrade.org/` and `http://www.fairtrade.ie/`), Silver Chilli (`http://www.silverchilli.com/`), Sustainable Harvest (`http://www.fairtrade.com/`). See especially: `http://www.fairtrade.net/`.

We emphasize that *we do not have, and the F&LT project does not have, any official connection with any of these organizations, nor are we endorsing them.* Instead, we are using this existing commercial niche for purposes of illustration. That said, we do note that part of our motivation with regard to this first goal of the project is to facilitate international trade by small and medium-sized enterprises (SMEs). The high startup cost for electronic trading under existing technologies, viz., EDI, is universally recognized as a major impediment to international trade by SMEs. The problem is well enough known to have a name (the first trade problem), and overcoming it is in large part what the U.N. means by the term *trade facilitation* (`http://www.unece.org/cefact/`).

Our second main goal in the F&LT project is to develop and implement a fundamental and comprehensive theory of business-related communication. Purchase orders, invoices, receiving reports, and so on—the previously-mentioned EDI documents—have been formalized often and in many ways, and continue to be so. Existing commercial standards, however, while improving their designs and embracing XML and other newer technologies, do little to manifest the underlying logical and semantic structures of their meanings [15]. In consequence, the *EDI mapping problem*—the problem of matching messages and their meanings with an organization's information processing systems—remains a costly challenge. Because there is no fundamental and principled way of interpreting the mes-

sages, or rather what they mean, we are left with an expensive ad hoc-racy, as is widely recognized and lamented.

We approach the problem of meaning representation for business-relation communication from a logicist perspective. On the face of it, prima facie, these messages are by nature propositional. One may capture them rather straightforwardly in ordinary language. The problem is to represent them formally in a way that is both reasonably complete (all the required information is captured) and that sanctions inferences correctly. The latter requirement is perhaps the strongest argument for logicism. A mere formalization, instantiated as a data structure, places no limits on what can be inferred. Valid and invalid inferences are all the same; a mere formalization does little to distinguish them. Thus a formalization by itself is insufficient for a fundamental theory, at least in this (broadly propositional) domain. If, for example, a valid purchase order is made and accepted, deontic states have changed and each party has new obligations and rights. Merely formalizing the information elements necessary for a purchase order, as is the object of most commercial messaging standards (X12, UN/EDIFACT, ebXML, etc.), fails entirely to address this deeper issue. And it is this issue in particular that is the focus of the second main goal of the F&LT project.[1]

The two goals are complementary. By 'light-weight business applications' we mean both comparatively simple systems implementations (goal 1), and comparatively limited and simple logico-semantic regimes. The strategy is to begin with relatively easy problems, to master them (conceptually and with fielded systems), and to generalize in the sequel.

The F&LT project grows out of a quite substantial stream of work, stretching back at least to the early 1980s (see [17] for discussion and extensive references). In what follows below, we can sketch only certain elements of the technical approach we are pursuing, and we continue to focus on the motivating factors for our technical approach. We begin, in the next section, with a characterization of *complex sentences* and a discussion of why they pose an immediate and crucial challenge to any logicist view of communication in business.

2 Problem: Understanding complex sentences

Linguists and grammarians have usefully distinguished three main types of sentences (e.g., [7, pages 142–4]).[2] First, a *simple sentence* consists of a single clause (verb plus noun phrase(s)) that "stands alone as a sentence." Roughly, a simple sentence corresponds to a predicate in first order logic with its arguments properly filled in. Second, a *coordinate sentence* is composed from simple sentences by such coordinating operators as *and, but,* and *or*. Roughly, in first order logic coordinate sentences are composed using the logical constants on predicates.

Third, a *complex sentence* is one in which "a clause can be incorporated *into* another clause" [7, page 143]. Davidson's famous example, "Galileo said that the earth moves" [6], serves to illustrate important distinctions for dealing with complex sentences. "the earth moves" is a clause (roughly, verb+noun phrase properly combined) that is here "incorporated" into the clause "Galileo said". The use of "that" is diagnostic of complex sentences; it typically serves to indicate the *subordinator* (subordinating clause, here "Galileo said") and the *subordinated* (or embedded) clause (here "the earth moves"). Adding to the relevant terminology, philosophers speak of an embedded clause (or the proposition it represents) as the *propositional content* of the embedding (complex) sentence. Also, often in practice "that" as a subordinate clause indicator is dropped, because no ambiguity results, as in "Galileo said the earth moves".

Linguistic complexity of this kind—sentences with embedded clauses, utterances that refer to other utterances—is quite common, and as noted is often signaled by the use of "that". Here is an example, drawn by convenience from fiction but quite representative of actual dialog.

> "There are too many other things that could have happened," I said. "That she did go away with Lavery and they split up. That she went away with some other man and the wire is a gag. That she went away alone or with a woman. That she drank herself over the edge and is holed up in some private sanatorium taking a cure. That she got into some jam we have no idea of. That she met with foul play."
>
> "Good God, don't say that," Kingsley exclaimed.
>
> – *The Lady in the Lake,* chapter 2, Raymond Chandler

Talk about sentences or propositions is complex in our present context (sentences with embedded clauses). It is ubiquitous and, in particular, it suffuses business messaging as well as legal discourse (see [8] for many examples). In addition, there is a well-recognized catalog of subordinating expression types (see [12] for a discussion), including those normally associated with these verbs: *believes* (as in *X believes that P*), *desires, intends,* and other mentalistic concepts; *asserts, promises, declares, commands,* and other speech acts; *it is necessary that, it is possible that, it is obligatory that, it is permitted that,* and other broadly

[1] There is insufficient space here for making good on the claim that our representational approach can capture, say, the deontic consequences of a promise or a purchase order. See [9] for a detailed treatment. There it is shown that Kimbrough's event semantics with disquotation approach can be made to work felicitously with Andrew J.I. Jones's multi-modal logic for conventional signalling systems.

[2] Some material in this section is edited from [13].

modal concepts; as well as such notions as *seeing to it that, feeling that, seeing that, hearing that, requiring that,* and *needing that*.

Complex sentences, utterances with embedded propositional content, pose considerable challenges to formalization in logic. First order logic does not permit, at least in any straightforward way, representation of embedded clauses. Moreover, first order logic is extensional and most complex sentences are intensional to some degree. Simplifying, we might express the generic complex sentence as having the form *P that Q,* where P is the subordinator, the embedding clause, and Q the subordinated, or embedded, clause (think: P = *Galileo said*; Q = *the earth moves*). In many cases, it may be true that *P that Q* and that $Q \leftrightarrow R$, but not that *P that R*. For example, let Q be *2+2=4* and R be *123-121=8-6*. Not only is it the case that $Q \leftrightarrow R$, but it is necessarily the case, yet it may well be true that Galileo said that Q and did not ever say that R. Similarly, it may be true that *P that Q(a)*, that $a = b$, and not true that *P that Q(b)*. Standard examples are created from a=*the morning star* and b=*the evening star*, and from a=*Cicero* and b=*Tully*. The morning star is, at it happens, one and the same as the evening star. *Cicero* and *Tully* are two names for the same historical individual. Also, we have Superman and Clark Kent, and Batman and Bruce Wayne, among others. Cases are unexotically generated with definite descriptions, even if one's favorite example is exotic: a=*Jocasta,* b=*the mother of Œdipus,* and c=*the wife of Œdipus.* (Mackie [20] contains a commendably clear and accessible treatment of these issues.)

3 Solution: Event semantics and disquotation

We are drawing on and developing an approach in first order logic that can represent and support inferencing on complex sentences, and in particular the kinds of complex sentences appearing in business communications.[3] It is called the *disquotational strategy for formalizing subordination* and it has begun to be explored (cf., [12, 14, 9, 18]). The key move in the disquotational approach is to treat a so-called quoted formula as a first order term and to provide meaning postulates that unpack—disquote—the embedded formula in order to associate it truth functionally with other formulas. The simplest of examples will suffice for present purposes. Let the intended interpretation of *Promise*(x, y, z) be *The event x is a promising by y of z*. For the particular case of s promising via event e that P we write *Promise*$(e, s, \lceil P \rceil)$. Associated with this is the meaning postulate (available as an axiom schema) *Promise*$(e, s, \lceil P \rceil) \rightarrow$ (*Kept*$(e) \leftrightarrow P$), which may be read as *If e is the event of s promising that P, then the promise named by e is kept if and only if P.*[4]

There is much to be said in principle in favor of a disquotation approach to representing embedded content. The papers cited above present evidence in this regard and prototype implementations have yielded positive results (e.g., [1, 2, 3, 21]). From a theoretical perspective, note that intensionality comes in degrees. The disquotation approach defaults to the strongest level of intensionality (substitution of terms of identical meaning is not guaranteed to preserve truth) and allows, via meaning postulates, arbitrary relaxation of intensionality. This affords the modeler great flexibility. From a practical perspective, note that finding and integrating the appropriate operator-based intensional logics may often be challenging—and the required logics may not exist. The disquotation approach affords an accessible, incremental approach to modeling and implementation. Unlike operator-based models, disquotational approaches allow individual obligation instances to be identified and referred to. This facilitates storing of deontic event histories, tracking of obligation life-cycles, and assignment of sanction in proportion to the severity of particular violations [4]. It is possible to identify some of the logical structure required and to implement it with the disquotation approach, leaving open the possibility of integrating its representations with operator-oriented logics. See [16] and [9] for exercises of this sort.

4 An example business scenario

The so-called *fair trade* movement has inspired a number of organizations to set up Web sites and serve as facilitators of transactions between developed-world buyers and developing-world suppliers. The fair trade facilitators serve, in part, by aggregating suppliers for better visibility and by helping to maintain the integrity of this form of small scale international trade. Typically, the purchases involve commodity agricultural products, such as coffee, or craft work, such as jewelry. Silver Chilli (`http://www.silverchilli.com/`) is a representative fair trade facilitator, which we shall use for purposes of illustration. We emphasize that we have no connection with Silver Chilli and are not in any way offering evaluative comment on it, positive or negative. Nor are we aiming to reverse engineer Silver Chilli's business or business systems. Instead, our suggestion is that transactions such as Silver Chilli now facilitates *could* be conducted with, and *could* benefit from,

[3] Some material in this section is edited from [13].

[4] Very many details are being suppressed, or elided, in the interests of focusing on the matters to hand. For example, a valid promise is arguably to the effect that one promises that *one sees to it that P*. Also, as emphasized by Searle [23, 24], promises are arguably forward-looking; what it is one promises must be something that occurs in the future, certainly not the past. We believe that the elisions committed are without prejudice to a complete and workable theory.

the sort of formalization and subsequent automation we discuss. Ultimately, of course, an adequate test of our ideas would require the implementation and successful fielding of such a system. For the present, however, that would be premature. Our aim here is to articulate certain general principles, using a particular example—Silver Chilli—for the sake of anchoring the discussion to its context of validity.

We begin with a simple, illustrative example, a successful purchasing transaction, and follow it beginning-to-end. There are six agents in this dialog: Buyer, a firm wishing to make a purchase; Seller, a firm from which Buyer wishes to make the purchase; TTP, a trusted third party facilitator or aggregator, through which the parties meet and the transactions are conducted (e.g., Silver Chilli); Shipper, a firm in the business of moving goods from sellers to buyers; B-Bank and S-Bank, Buyer's bank and Seller's bank respectively.

Preparatory to the interaction, both Buyer and Seller have made contractual agreements with TTP. Further, the Seller has posted a catalog and a menu of shipping options via TTP. Informally, the transaction dialog proceeds as shown in the following steps:

A. Buyer: Views Seller's catalog and shipping options and decides to make a purchase.

B. Buyer: Formulates a purchase order and sends it to Seller via TTP.

C. TTP: Receives the purchase order message from Buyer, records it, and forwards it to Seller.

D. Seller: Receives purchase order, decides to honor it, sends a message to this effect to Buyer via TTP.

E. TTP: Receives the purchase order acceptance message from Seller, records it, and forwards it to Buyer.

F. Buyer: Receives and records purchase order acceptance message.

G. Seller: Commences shipping process.

H. Shipper: Receives goods and shipping documents from Seller; starts the delivery process.

I. Seller: Notifies Buyer, via TTP, that shipment has been made, and provides details of the shipping transaction.

J. TTP: Receives shipping notification message, records it, and forwards it to Buyer who records it.

K. Shipper: Presents goods for delivery to Buyer.

L. Buyer: Inspects goods, matches delivery with purchase order, elects to accept delivery as complete, sends acceptance message to Seller via TTP.

M. TTP: Receives acceptance message, records it, authorizes execution of payment by Buyer's Bank to Seller's Bank, and notifies Seller.

N. Buyer: Directs Buyer's bank to pay Seller, and sends notification of this to Seller via TTP.

O. TTP: Receives payment notification, records it, and forwards it to Seller.

We will now discuss each of the steps in the transaction dialog individually.

4.1 Buyer: Views Seller's catalog and shipping options and decides to make a purchase.

Silver Chilli's Web page catalog for bracelets will serve to anchor the example. At the time of writing, it has six entries, each entry having only a few fields of information:

1. Name of item, presumably a unique name which can be used as a key.

   Examples: ayaulta bracelet and tejida plata bracelet

2. Text describing the article.

   Examples: A taste of spring! Delicate daisies suspended between fine shafts of silver make a truly delicate neck piece designed to finish any outfit. and A stunningly elegant solid silver design to adorn your wrist on every occasion. The delicate weave pattern is a reminder of the great skill and precision of our silversmiths.

3. Currency in use.

   British pounds in each case: GBP.

4. Unit price (in the currency in use).

   E.g., 15.00 and 35.00

It would be natural and quite unproblematic to represent this information (and catalog information more generally) in a relational database, whose structure could be published and for which clients (e.g. Buyer and Seller) could be given read-only access. We note, however, that relational records are also translated into first-order logic quite unproblematically and it will be useful conceptually, in what follows, to think of them so translated. In the simple case to hand, we may consider the catalog as a single relation, called catalog, having four attributes: itemid, description, currency, and unit_price. An example record, translated into first order logic, could be expressed as

Logic Expression 1 catalogItem('ayaulta bracelet', 'A taste of spring! Delicate daisies suspended between fine shafts of silver make a truly delicate neck piece designed to finish any outfit.', 'GBP', 15.00)

Further, we assume that a number of shipping options are posted, e.g., air express by FedEx, each of which has an ID and a description. For the sake of simplicity we do not model this feature of the transaction.

4.2 Buyer: Formulates a purchase order and sends it to Seller via TTP.

Buyer, let us assume, decides to purchase 11 of the ayaulta bracelets, at GBP15.00 each plus the cost of shipping by FedEx AirFreight. Buyer then formulates a purchase order and sends it to Seller, via the TTP.

FLBC Message 1 (A Purchase Order)
po(*b:e1*) ∧ *Speaker*(*b:e1, Buyer*) ∧ *Addressee*(*b:e1, Seller*) ∧ *Cul*(*b:e1, t1*) ∧ *S-Content*(*b:e1,* ⌈*deliver*(*b:e2*) ∧ *Agent*(*b:e2, Seller*) ∧ *Benefactive*(*b:e2, Buyer*) ∧ *Cul*(*b:e2, b:t2*) ∧ *b:t2* ≤ *t1+30* ∧ *Theme*(*b:e2, 'ayaulta bracelet'*) ∧ *Unit*(*'ayaulta bracelet', item*) ∧ *Quantity*(*'ayaulta bracelet', 11*) ∧ *shipping_mode*(*'FedEx AirFreight'*) ∧ *Cost_for*(*b:e2, 'FedEx AirFreight', b:c1*)⌉) ∧ *B-Content*(*b:e1,*⌈*pay*(*b:e3*) ∧ *Agent*(*b:e3, Buyer*) ∧ *Benefactive*(*b:e3, Seller*) ∧ *Sake*(*b:e3, b:e2*) ∧ *Theme*(*b:e3, 'GBP'*) ∧ *Unit*(*'GBP', item*) ∧ *Quantity*(*'GBP', 165.00+b:c1*) ∧ *Cul*(*b:e3, b:t3*) ∧ *b:t3* ≤ *b:t2+15*⌉)

A word on notation. Each of the terms present in FLBC Message 1 are to be understood as individual constants. The colon notation— '*:*'—indicates namespace. Thus, for example, *b:e2* is sequence name *e2* from *b*'s namespace, i.e., the namespace of Buyer. Unquoted names without attendant namespace indicators, e.g., *t1*, are to be understood as shorthand for fully articulated names. In this case, *t1* would be short for a specified date, for example 14 July 2006. The Cul predicate (culmination) indicates that the speaker chooses to perform the utterance connected to the first predicate argument (a message ID), at the time given in the second argument (for further discussion of the Cul predicate see [22, 16]).

Note that some times are qualified by namespace indicators, e.g., *b:t2*, in which case the full expression is simply another term in the namespace sequence. Note, for example, that *b:c1* is a *name* for the shipping for the requested delivery of goods; it is not, however, a number indicating the cost. That number will be determined later, once it is known.

Points arising:

1. Purchase order—*po*—is represented as a basic communicative act, with *two* embedded (and quoted) content clauses. The first, *S-Content,* describes a delivery event, roughly Seller's seeing to it that the indicated goods are delivered to Buyer. The second, *B-Content,* describes a payment by Buyer to Seller, which is for the sake of—alternatively put, is consideration for—the delivery just described.

   Notice that nothing is said explicitly about transference of ownership of the goods delivered. Different legal systems may have different conventions about when it is that ownership changes. Representation of these conventions belongs in the governing interchange agreement. Forming the interchange agreement contributes much of the expense involved in electronic trading under EDI (and its variants). The F&LT project may be seen, in no small part, as an effort to formalize logically much of the interchange agreement. For example, the rule may be that if a purchase order is issued and accepted, and if the stated delivery occurs, then ownership is transferred to the buyer upon delivery.

2. A purchase order may be seen as a type of request. The Speaker is requesting that a certain commercial transaction take place. We use Jones's Request Schema (Expression 1) to govern all requests.

   Expression 1 (Jones's Request Schema)
   $E_jU \Rightarrow_s I^*_s H_j E_k C$

   The antecedent of the $\Rightarrow_s$ conditional, E_jU, describes j's act of performing an act which, by convention in signalling system s, is of type 'request'. The consequent says that, relative to the function, or purpose, that signalling system s is supposed to fulfil, the ideal upshot of the performance of the act of requesting is that the requester, j, is attempting to see to it (H_j) that the requestee, k, sees to it that C (E_kC). For a detailed account of the relativised optimality operator I^*_s, and of the rôle it plays in the specification of conventions for a range of different types of communicative acts, see [9] and [11]. $\Rightarrow_s$, relativised to conventional signalling system s, is the 'counts as' operator of Jones and Sergot [10]. Further specification is required for purchase orders:

   Axiom Schema 1 (Governing Request-PO)
   $E_j(po(e) \wedge Speaker(e,j) \wedge Addressee(e,k) \wedge Cul(e,t) \wedge S\text{-}Content(e, \lceil C \rceil)) \Rightarrow_f I^*_f H_j E_k C$

   Axiom Schema 1 is, we assume, part of the interchange agreement and is available to all parties for purposes of inferencing.

3. We also take the view, for this system, that to issue a purpose order is to issue a request that it be acknowledged, positively or negatively, within say 14 days. That rule, again, belonging to the interchange agreement is as follows.

Axiom Schema 2 (Governing Request-PO-Reply)
$E_j(po(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Cul(e, t)) \Rightarrow_f I^*_f OE_k(reply(e1) \wedge Speaker(e1, k) \wedge Addressee(e1, k) \wedge Theme(e1, e) \wedge Cul(e1, t1) \wedge t1 \leq t+14)$

4.3 TTP: Receives the purchase order message from Buyer, records it, and forwards it to Seller.

TTP's rôle here is largely passive. It receives the message, Logic Expression 1, validates it, records it, and forwards it to Buyer. The validation process may also be seen as deductive and based on validation rules possessed by TTP. These would include rules to determine whether Logic Expression 1 is well-formed, whether Buyer is permitted to issue purchase orders, and whether Seller is permitted to receive them. The extent of the validation process would be determined by application-specific conditions. We note that the fact—that the message, the purchase order in this case, is so transparently propositional—facilitates a rule-based approach to validation, with all its attendant advantages.

4.4 Seller: Receives purchase order, decides to honor it, sends a message to this effect to Buyer via TTP.

Having decided that the purchase order is in order and that it wishes to do business accordingly, Seller accepts the purchase order by sending FLBC Message 2 to Buyer, via TTP.

FLBC Message 2 (Seller Accepts Purchase Order)
$accept(s{:}e1) \wedge Speaker(s{:}e1, Seller) \wedge Addressee(s{:}e1, Buyer) \wedge Theme(s{:}e1, b{:}e1) \wedge Cul(s{:}e1, now)$

4.5 TTP: Receives the purchase order acceptance message from Seller, records it, and forwards it to Buyer.

Axiom Schema 2, above, prescribes that purchase orders should be (ideally are) replied to in a timely manner. Axiom Schema 3, another general rule belong to the interchange agreement, states that accepting a purchase order entails replying to it. Note, however, that accepting does not entail meeting the requirement specified by Axiom Schema 2, since the acceptance may be tardy. Thus, the axiom schemas support a distinction between whether a proper reply has been made and whether any reply at all has been made.

Axiom Schema 3 (To Accept Is to Reply)
$E_j(accept(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Theme(e, e1) \wedge po(e1) \wedge Speaker(e1, k) \wedge Addressee(e1, j) \wedge Cul(e, t)) \rightarrow E_j(reply(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge Theme(e, e1) \wedge Cul(e, t))$

More importantly, it is now the case that a purchase order is in effect. Seller has promised to see to it that a delivery is made and Buyer has promised to pay for it. Again, a general rule belonging to the interchange agreement capture this.

Axiom Schema 4 (PO Commitments)
$\{po(e) \wedge Speaker(e, j) \wedge Addressee(e, k) \wedge S\text{-}Content(e, \lceil C \rceil) \wedge B\text{-}Content(e, \lceil D \rceil) \wedge accept(e1) \wedge Speaker(e1, k) \wedge Addressee(e1, j) \wedge Theme(e1, e) \wedge Cul(e2, t)\} \rightarrow \{commit(e2) \wedge Speaker(e2, k) \wedge Addressee(e2, j) \wedge Content(e2, \lceil C \rceil) \wedge Cul(e2, t)\} \wedge \{commit(e3) \wedge Speaker(e3, j) \wedge Addressee(e3, k) \wedge Content(e3, \lceil D \rceil) \wedge Cul(e3, t)\}$

Essentially, Axiom Schema 4 states formally that if j offers a purchase order to k and k accepts, then k is committed to delivering the goods to j and j is committed to paying for them. A few points by way of comment.

1. According to Axiom Schema 4, the mentioned commitments by buyer and seller come into effect at time, t, the time of utterance of a valid accepting of the purchase order by the seller. Other policies are possible and there is nothing in the logic that mandates this particular policy. We simply present it as a plausible example.

2. Most interestingly, perhaps, note that the resulting commitments each have a new unique identifier, *e2* for the seller's commitment and *e3* for the buyer's. These communicative acts may properly be said to be implicit. Neither the buyer nor the seller ever says explicitly anything like "I promise that…" Instead, the transaction dialog is conducted under conventions that, embodied in the various axiom schemas discussed above, license the inference that a promises has been made.

4.6 Buyer: Receives and records purchase order acceptance message.

There is little for Buyer to do in the present simple example. We note that were Seller to provide additional information, e.g., on expected timing of the shipment, Buyer could use it to infer corresponding actions, perhaps for planning purposes.

4.7 Seller: Commences shipping process.

Seller must contact Shipper and arrange for pickup and delivery of the goods in question. Complications will arise because our example assumes international trade. Clearly, this side aspect of the full transaction could benefit from the principles under discussion for the main scenario. In the interests of simplicity, however, we leave this for another time and place. Seller and Shipper may use any available means to see to it that the goods are delivered as promised.

4.8 Shipper: Receives goods and shipping documents from Seller; starts the delivery process.

This step is inherently physical and only minimally informational. The shipper must take physical possession of the goods and convey them to the buyer. This process may be aided by the principles under discussion here, but again we leave the details for another venue.

4.9 Seller: Notifies Buyer, via TTP, that shipment has been made, and provides details of the shipping transaction.

With the goods dispatched to Shipper, Seller notifies Buyer that the shipment has commenced and what the expected delivery date is. In terms of communicative acts, Seller is making two assertions: that the shipment is underway and that the shipment is expected to arrive on a certain date.

FLBC Message 3 (Shipping Notice)
assert(*s:e2*) ∧ *Speaker*(*s:e2, Seller*) ∧ *Addressee*(*s:e2, Buyer*) ∧ *Cul*(*s:e2, t5*) ∧ *Content*(*s:e2,* ⌈ *shipping*(*s:e3*) ∧ *Experiencer*(*s:e3, b:e2*) ∧ *Benefactive*(*s:e3, Buyer*) ∧ *Hold*(*s:e3, now*) ∧ *Start*(*s:e3, t5*) ∧ *TrackingNumber*(*s:e3, a1*)⌉)

Points arising:

1. Shipping is here understood to be a process that extends over time. The message is presumed to be sent after shipping has begun and before it has ended.

2. *TrackingNumber* maps the seller's shipping ID with the shipper's ID for this transaction.

4.10 TTP: Receives shipping notification message, records it, and forwards it to Buyer who records it.

TTP's rôle here is passive, although it does perform validation on the message before storing it and forwarding it to Buyer.

4.11 Shipper: Presents goods for delivery to Buyer.

Shipper for its own purposes will record the event and will require authentication from Buyer and acknowledgement of receipt. Again, although the principles and techniques under discussion here are relevant to this aspect of the transaction, we skip over them in the interests of brevity.

4.12 Buyer: Inspects goods, matches delivery with purchase order, elects to accept delivery as complete, sends acceptance message to Seller via TTP.

Buyer's receiving agent will be empowered to inspect Shipper's manifest and the goods upon delivery in order to determine whether to accept them. The shipper's manifest should be numbered *a1* (the *TrackingNumber,* see FLBC Message 3). Assuming everything appears to be in order, Buyer's receiving agent will 'sign for' the shipment, acknowledging receipt of the goods from Shipper. Then, in accordance with governing rules in the interchange agreement, Buyer sends a message to Seller, via TTP, acknowledging receipt.

FLBC Message 4 (Receipt Acknowledgment)
accept(*b:e4*) ∧ *Speaker*(*b:e4, Buyer*) ∧ *Addressee*(*b:e4, Seller*) ∧ *Theme*(*b:e4, b:e5*) ∧ *S-Content*(*b:e1, b:e5*) ∧ *Cul*(*b:e4, now*)

There is a bit of subtlety here. Buyer accepts *b:e5*, the referent of a new unique name. By conjoining ∧ *S-Content*(*b:e1, b:e5*) in FLBC Message 4, Buyer fixes the reference of *b:e5* to the (quoted) description of Seller's obligation in the original purchase order. This is merely shorthand, for convenience. Instead, Buyer could have said:

FLBC Message 5 (Receipt Acknowledgment—longhand)

accept(*b:e4*) ∧ *Speaker*(*b:e4, Buyer*) ∧ *Addressee*(*b:e4, Seller*) ∧ *Theme*(*b:e4,* ⌈*deliver*(*b:e2*) ∧ *Agent*(*b:e2, Seller*) ∧ *Benefactive*(*b:e2, Buyer*) ∧ *Cul*(*b:e2, b:t2*) ∧ *b:t2* ≤ *t1+30* ∧ *Theme*(*b:e2, 'ayaulta bracelet'*) ∧ *Unit*(*'ayaulta bracelet', item*) ∧ *Quantity*(*'ayaulta bracelet', 11*) ∧ *Shipping_Mode*(*b:e2, 'FedEx AirFreight'*) ∧ *Cost_for*(*b:e2, 'FedEx AirFreight', b:c1*)⌉) ∧ *Cul*(*b:e4, now*)

Notice as well, in either case, that Buyer does not *assert* that C, rather Buyer *accepts* that C. Buyer is not saying that C happened exactly as described; instead, Buyer is saying he declares, so far as Buyer is concerned, that C, which is Seller's obligation in the transaction, has been satisfied. The goods may in fact have arrived late, and not as described in C. Buyer is anyway declaring himself content with the delivery.

4.13 TTP: Receives acceptance message, records it, and forwards the message to Seller.

Recording and forwarding are, as before, essentially passive (except for validation by TTP). Are there any interesting new inferences to be made? As discussed above, even before the sending of FLBC Message 4, TTP is licensed to deduce that $I_f^* O E_j C$ and $I_f^* O E_k D$, with j instantiated to Seller, C instantiated to the content Seller's commitment, k instantiated to Buyer, and D to the content of Buyer's commitment. Buyer has now, with FLBC Message 4, declared that C. Are we ready to conclude that C? If we are, the logical tools to hand allow us to express the policy rule that would warrant the inference. We discuss this further in the next section.

4.14 Buyer: Directs Buyer's bank to pay Seller, and sends notification of this to Seller via TTP.

For the usual reasons, we defer modelling of the bank interactions. Having directed payment to Seller, Buyer sends a confirmation to Seller via TTP:

FLBC Message 6 (Confirm Payment)
assert(*b:e5*) $\wedge$ *Speaker*(*b:e5, Buyer*) $\wedge$ *Addressee*(*b:e5, Seller*) $\wedge$ *Theme*(*b:e5, b:e6*) $\wedge$ *B-Content*(*b:e1, b:e6*) $\wedge$ *Cul*(*b:e5, now*)

4.15 TTP: Receives payment notification, records it, and forwards it to Seller.

Upon validating the message, TTP stores it and forwards it to Seller, who can be expected to monitor it bank account for timely arrival of the payment.

5 Deductions over traces using our representation

In order to effect FLBC interactions in electronic commerce, the logic-based forms of the FLBC messages presented in previous sections must be transformed to a software form. For our early experiments we have used Prolog as our deductive database—the conjunctive form used in the logic-based FLBC representations maps particularly neatly into Prolog syntax. For example, the initial parts of the Prolog form of FLBC Message 1 is shown in figure 1.

The `mdb` terms are used represent FLBC knowledge. The 3-tuple form we have used almost everywhere is purposefully similar to knowledge representation using the Resource Description Framework (RDF, see [26]). It is intended future work to collect the FLBC predicates into a

```
mdb(b:e1,nType,po).
mdb(b:e1,speaker,buyer).
mdb(b:e1,addressee,seller).
mdb(b:e1,cul,b:t1).
mdb(b:t1,value,Now).
mdb(b:e1,content,s,b:e2).
mdb(b:e1,content,b,b:e3).

mdb(b:e2,nType,deliver).
mdb(b:e2,agent,seller).
mdb(b:e2,benefactive,buyer).
mdb(b:e2,cul,b:t2).
mdb(b:e2,require,b:e2-r1):-
        mdb(b:t1,value,T1),
        mdb(b:t2,value,T2),
        T2 =< T1 + 30.
```

Figure 1. Initial clauses of FLBC Message 1 shown using our Prolog representation.

published RDF ontology. Doing so will allow us to participate to some degree in the Semantic Web, at least with respect to taking advantage of its increasing tool set.

As discussed in section 4.2, the syntax `TermA:TermB` indicates our representation of namespace scoping—`TermB` is a name 'owned' by `TermA`. We assume that this owner is capable of keeping their namespace in order, and thus effect distributed name management. We have used a different connective '`-`' to allow hierarchical namespace management. For example the name `b:e3-amount` allows the owner `b` of `e3` to indicate that this `amount` is strongly related to `e3`, avoiding the need to populate `b`'s top-level namespace with single-use values.

Generally our knowledge terms are of the form `mdb(X,Y,Z)` that describes the predicate `Y` to be `Z` when operating on subject `X`. This is almost exactly how predicates operate in RDF. Note, however, that we deviate in two respects:

4-tuples. Certain FLBC predicates, for example the `content` of purchase order messages, actually use a 4-tuple form. In RDF the need to represent more than one object of a particular predicate is handled through the use of anonymous intermediate nodes. We could easily have adopted this strictly 3-tuple form, but for unification becoming more awkward.

Clauses with bodies. Instances of one particular FLBC predicate, `require`, are defined with associated clause bodies. RDF is data-oriented rather than being computational, thus there is no direct parallel. Being able to store 'programs' that perform data checks alongside the data they will be validating is something Prolog does particularly well.

In figure 1, the name `b:e1` represents FLBC Message 1. For ease of reference, we define `nType` predicate instances to indicate the type of the message formed from all the predicates for which `b:e1` is a subject. An RDF version of our representation would indicate that the `nType` predicate itself was 'owned' by a central FLBC namespace. As much as possible the meaning of the Prolog-based FLBC predicates is unchanged from that discussed in previous sections.

Disquotation is effected by including FLBC Message names as the objects of predicates. For example, `mdb(b:e3, sake, b:e2).` indicates that message `b:e3`'s `sake` predicate disquotes message `b:e2`. FLBC variables are described and referenced in the same manner.

FLBC variables such as the time `b-t1` will often be referred to by name (of course they are 'facts' from the Prolog perspective). This is necessary to allow restrictions to be defined and exchanged between parties before the actual values of such FLBC variables have been bound. FLBC variables are bound using the `value` FLBC predicate.

Our actual database contains numerous additional predicates used to provide descriptions in diagnostic output that we have omitted for the sake of brevity.

5.1 Implementation

Having settled on a Prolog representation, we developed a simple distributed communications framework so that we could run the example interaction detailed in section 4.

With an eye to planned future research, our message delivery infrastructure is actually a basic publish/subscribe system. A 'Communications Manager' process is responsible for brokering message delivery, and thus is ideally placed to fulfil the rôle of the TTP. Admittedly for our tests the Communications Manager was not FLBC-aware. Making it so will be trivial, however, since its only extra jobs will be caching FLBC knowledge and repeating the requirements checks already being done by the other interacting parties.

Our example scenario involves the interaction of a buyer 'b' and a seller 's'. These two parties were run on different Prolog processes (with independent clause databases). The communication was facilitated by assigning each process a particular TCP/IP port. Note that all these ports were on the same physical computer—this may preclude certain concurrency conditions, however our results will not be affected given the 's'/'b' interaction is a serialised, reactive. dialogue. We are investigating use of the Event Calculus or other verification formalisms to enable us to attempt any required correctness proofs for more complex concurrent multi-party interactions.

Message transmission involves each FLBC message being scanned for dependencies, and all referenced terms also being sent over to the destination party. We intend to specify the interchange agreement between parties in terms of FLBC predicates. When this is done, it will provide a list of terms to be dropped from dependency analysis—it is known that both parties already know them (or would request them from the TTP on demand). The message transmission code knows that not all terms will be defined at the time a message is sent. In our scenario, term `b:t2`—to be the time of the delivery—is a dependency at the time term `b:e1` is transmitted, but has not yet been given a value.

```
Run steps [1,2 3,4,5] of CoALa FALT scenario.
[T=0 1: Buyer creates and sends purchase order]
sendFLBC: checking requirements...
sendFLBC: ...done. Dependencies defined ATM:
  {b:e1, b:e2, b:e3, b:t1, b:e2-ship,
   b:e2-toBuy, b:e3-amount}
sendFLBC: Sending root [b:e1] to s from b.
[T=2 2: Seller accepts purchase order]
[T=12 3: Seller sends shipping notice]
[T=13 4: Buyer acknowledges receipt]
sendFLBC: checking requirements...
ttp: check b:e2 requirements: passed:
   Require delivery within 30 days
sendFLBC: ...done. Dependencies defined ATM:
  {b:c1, b:e2, b:e4, b:t2, b:e2-r1,
   b:e2-ship, b:e2-toBuy}
sendFLBC: Sending root [b:e4] to s from b.
[T=14 5: Buyer confirms payment]
sendFLBC: checking requirements...
ttp: check b:e3 requirements: passed:
   Payment must be within 15 days of delivery.
sendFLBC: ...done. Dependencies defined ATM:
  {b:c1, b:e2, b:e3, b:e5, b:t2, b:t3,
   b:e2-r1, b:e2-ship, b:e2-toBuy, b:e3-amount,
   b:e3-payment, b:e3-r1, b:e3-amount-r1}
sendFLBC: Sending root [b:e5] to s from b.
```

Figure 2. Event trace diagnostic output from the Buyer's perspective

We verified that our framework was producing expected results by first running both processes without real message transmission. After the trace completed 'b', contained only `b-_` predicates, and 's' only `s-_` ones. Enabling communication resulted in both having a complete copy of the FLBC predicates.

5.2 Hypothetical actions

The reader will note from the last section that there may be redundant copies of data created during any multi-party interaction. We argue that this redundancy in fact has many benefits—should it become a problem with respect to storage space, we could use the TTP as a reference point providing authoritative versions of some subset of the FLBC predicates defined for a given dialogue.

One important aspect of storing an event from both per-

```
Run steps [1,2 3,4,5] of CoALa FALT scenario.
[T=0 1: Buyer creates and sends purchase order]
[T=2 2: Seller accepts purchase order]
sendFLBC: checking requirements...
sendFLBC: ...done. Dependencies defined ATM:
  {s:e1}
sendFLBC: Sending root [s:e1] to b from s.
[T=12 3: Seller sends shipping notice]
sendFLBC: checking requirements...
sendFLBC: ...done. Dependencies defined ATM:
  {s:c1, s:e2, s:e3, s:t5}
sendFLBC: Sending root [s:e2] to b from s.
[T=13 4: Buyer acknowledges receipt]
[T=14 5: Buyer confirms payment]

Checking all requirements at end of transaction.
ttp: check s:e1 requirements: none.
ttp: check s:e2 requirements: none.
ttp: check s:e3 requirements: none.
```

Figure 3. Event trace diagnostic output from the Seller's perspective

```
Run steps [1,2 3,4,5] of CoALa FALT scenario.
[T=0 1: Buyer creates and sends purchase order]
sendFLBC: checking requirements...
sendFLBC: ...done. Dependencies defined ATM:
  {b:e1, b:e2, b:e3, b:t1, b:e2-ship,
   b:e2-toBuy, b:e3-amount}
sendFLBC: Sending root [b:e1] to s from b.
[T=2 2: Seller accepts purchase order]
[T=32 3: Seller sends shipping notice]
[T=33 4: Buyer acknowledges receipt]
sendFLBC: checking requirements...
ttp: check b:e2 requirements: FAILED:
   Require delivery within 30 days
```

Figure 4. Event trace diagnostic output for a Buyer discovering an unfulfilled requirement

spectives is to allow conflict resolution. For example, if `a:e120` was a speech act indicating 'a's belief that 1,000 pencils had been delivered, whereas `b:e120` might indicate 'b's belief that 999 pencils had been delivered. It can then be explicitly stated and recorded that 'b' considers that `b:e120` counts as `a:e120`.

Another important aspect is facilitating prospective experimentation—there is no need to make a strong differentiation between terms created through interchange with another party, and terms asserted locally. Attempts to forward clauses based on local assertions would fail when the TTP is unable to verify the state of affairs reached. If a party keeps track of their local assertions, however, a rich environment is provided in which they can experiment with hypothetical situations.

Figures 2 (from the buyer's perspective) and 3 (from the seller's perspective) show diagnostic output from an example successful interaction between the parties. For lines contained within square brackets, the `T` value is 'time', and the number directly preceding the colon is the step number. The rest of the line is a brief description of the function of that step. Note that no action is actually taken unless this description line is directly followed by one or more lines with the prefix `sendFLBC:`. The set of FLBC names shown in braces is the result of performing transitive closure over dependencies (that are defined at that moment) of the event being sent. Any requirements which need to be checked will have diagnostics presented on lines with the prefix `ttp:`.

By way of contrast, an event trace of the buyer discovering a failed requirements check is shown in figure 4.

6 Discussion

This project builds upon an extensive stream of directly-related work, portions of which have been cited above. Even so, there are very many areas of this project which require a great deal of further work. We indicate here just a few of these areas. Any specific software system for supporting will at best support a limited number of types of trading scenarios. It is imperative that these types be fully articulated and validated both legally (Are the arrangements adequate from a legal perspective? Could a competent attorney advise a client to trade under them?) and commercially (Are the arrangements workable and suitable for 'real-world' use?). In terms of our logic framework, we need to ensure we have sufficient formalisation of interchange agreements to allow the trusted third parties to perform rich forms of validation and inferencing on the FLBC messages. To this end we plan, among other things, to layer on a form of the event calculus (cf., [19] and later versions). We believe this will fit naturally and most usefully with the FLBC event semantics and disquotation formalism we now use. On the technical side, we need to assess the most ap-

propriate mechanism to effect the transmission and storage of locally-created terms and predicates between the parties in our business interactions.

7 Conclusion

In conclusion, we have presented the goals of the Fair and Logical Trade project. We believe our logic-based approach to business communication will allow a far larger number of individuals and organizations to participate in electronic commerce. Our incremental approach to formalization allows us to support a wide variety of scenarios—at one end of the spectrum providing compositional term-sets to describe simple business interactions, and at the other developing comprehensive interchange agreements to provide explicit semantics for the deontic notions described in the business exchange messages themselves. The F&LT project draws upon a body of research in which many of the logical problems are raised and addressed satisfactorily. The real test of this, of course, lies in successful reduction to practice, which is a prime motivator for the project. In this paper, we have introduced the F&LT project and reported on our first new results, presenting elements from an early prototype, and outlining the many areas of future work remaining. We hope others will find this a promising enterprise and we invite those who do to participate.

Acknowledgment

David Eyers is supported by EPSRC grant GR/S94919/01.

References

[1] A. Abrahams. *Developing and Executing Electronic Commerce Applications with Occurrences*. PhD thesis, University of Cambridge Computer Laboratory, 2002.

[2] A. S. Abrahams and J. M. Bacon. A software implementation of Kimbrough's disquotation theory for representing and enforcing electronic commerce contracts. *Group Decision and Negotiations Journal*, 11(6):1–38, November 2002.

[3] A. S. Abrahams, D. M. Eyers, and J. M. Bacon. Practical contract storage, checking, and enforcement for business process automation. In S. O. Kimbrough and D. J. Wu, editors, *Formal Modelling for Electronic Commerce*, International Handbooks on Information Systems, pages 33–77. Springer, Berlin, Germany, 2004.

[4] Alan S. Abrahams and Jean M. Bacon. The Life and Times of Identified, Situated, and Conflicting Norms. In *Proceedings of the 6th International Workshop on Deontic Logic in Computer Science (DEON'02)*, pages 3–20, London, England, May 2002.

[5] D. Davidson. On saying that. *Synthèse*, 19:130–46, 1968–9.

[6] D. Davidson. *Inquiries into Truth and Interpretation*, chapter On Saying That, pages 93–108. Clarendon Press, Oxford University Press, Walton Street, Oxford 0X2 6DP, United Kingdom, 1984. Originally published as [5].

[7] E. Finegan. *Language: Its Structure and Use*. Harcourt Brace College Publishers, Fort Worth, TX, third edition, 1999.

[8] A. v. Gardner. *An Artificial Intelligence Approach to Legal Reasoning*. Artificial Intelligence and Legal Reasoning. MIT Press, Cambridge, MA, 1987.

[9] A. J. Jones and S. O. Kimbrough. A note on modelling speech acts as signalling conventions. In S. O. Kimbrough and D. J. Wu, editors, *Formal Modelling in Electronic Commerce*, International Handbooks on Information Systems, pages 325–342. Springer, Berlin, Germany, 2004. ISBN 3-540-21431-3.

[10] A. J. Jones and M. J. Sergot. A formal characterisation of institutionalised power. *Journal of the Interest Group in Pure and Applied Logic (IGPL)*, 4(3):427–443, 1996. Reprinted in [25, pages 349–367].

[11] A. J. I. Jones and X. Parent. Conventional signalling acts and conversation. In F. Dignum, editor, *Advances in Agent Communication*, volume 2922, pages 1–17, Berlin, 2004. Springer-Verlag.

[12] S. O. Kimbrough. Reasoning about the objects of attitudes and operators: Towards a disquotation theory for representation of propositional content. In *Proceedings of ICAIL '01, International Conference on Artificial Intelligence and Law*, 2001.

[13] S. O. Kimbrough. A note on interpretations for federated languages and the use of disquotation. In A. Gardner, editor, *Proceedings of the Tenth International Conference on Artificial Intelligence and Law (ICAIL-2005)*, pages 10–19, Bologna, Italy, June 6–11, 2005. In cooperation with ACM SIGART and The American Association for Artificial Intelligence.

[14] S. O. Kimbrough. A note on the Good Samaritan paradox and the disquotation theory of propositional content. In J. Horty and A. J. Jones, editors, *Proceedings of ΔEON'02, Sixth International Workshop on Deontic Logic in Computer Science*, pages 139–148, May 2002.

[15] S. O. Kimbrough and S. A. Moore. On obligation, time, and defeasibility in systems for electronic commerce. In J. F. Nunamaker, Jr. and R. 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.

[16] S. O. Kimbrough and Y.-H. Tan. On lean messaging with unfolding and unwrapping for electronic commerce. *International Journal of Electronic Commerce*, 5(1):83–108, 2000.

[17] S. O. Kimbrough and D. J. Wu. FMEC: Overview and interpretation. In S. O. Kimbrough and D. J. Wu, editors, *Formal Modelling for Electronic Commerce*, pages 1–29. Springer, Berlin, Germany, 2004.

[18] S. O. Kimbrough and Y. Yang. On representing special languages with flbc: Message markers and reference fixing in

seaspeak. In S. O. Kimbrough and D. J. Wu, editors, *Formal Modelling in Electronic Commerce*, International Handbooks on Information Systems, pages 297–324. Springer, Berlin, Germany, 2004. ISBN 3-540-21431-3.

[19] R. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing, Springer Verlag*, 4:67–95, 1986.

[20] J. L. Mackie. Problems of intentionality. In Joan and P. Mackie, editors, *Logic and Knowledge: Selected Papers, Volume I*, pages 102–116. Oxford University Press, Oxford, England, 1985. ISBN: 0 19 824679 X.

[21] S. A. Moore. *Saying and Doing: Uses of Formal Languages in the Conduct of Business*. PhD thesis, University of Pennsylvania, The Wharton School, Philadelphia, PA, 19104, USA, December 1993.

[22] T. 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.

[23] J. R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, England, 1969.

[24] J. R. Searle and D. Vanderveken. *Foundations of Illocutionary Logic*. Cambridge University Press, Cambridge, England, 1985.

[25] E. G. Valdés et al., editors. *Normative Systems in Legal and Moral Theory – Festschrift for Carlos E. Alchourrón and Eugenio Bulygin*. Duncker & Humblot, Berlin, Germany, 1997.

[26] W3C. Resource description framework (RDF) model and syntax specification. `http://www.w3.org/TR/rdf-primer/`, Feb. 1999.