Using Event Semantics for Modeling Contracts

Yao-Hua Tan
Dept. of Economics and Business Administration
Free University Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
ytan@feweb.vu.nl

Walter Thoen
EURIDIS
Erasmus University Rotterdam
Burgemeester Oudlaan 50, 3062 PA Rotterdam
wthoen@fac.fbk.eur.nl

Abstract

Currently a number of these on-line support systems for electronic contracting are under development. In this paper we develop a logical formalism to represent the content of business contracts. This formalism can be used to develop applications that can automatically negotiate and process contracts, or it can be used to develop on-line help systems that explain to the human negotiator what for him the implications are of a certain contract that is proposed to him by the counter-party. The formalism we develop is based upon recent developments in the field of the Formal Language for Business Communication (FLBC) and event semantics. In the paper we show how many key constructs of the content of business contracts can be modeled using event semantics. We also show that the formalism can be implemented in Prolog.

1. Introduction

In business-to-business on-line community building it is essential that companies cannot only meet on-line, but also establish business relationships on-line by negotiating contracts on-line. In other words, we view electronic contracting as an important infrastructure enabler for on-line community building. The concept of 'electronic contracting' was first introduced over a decade ago by Ronald Lee [11][12][13]. Lee referred to the language and formal procedures of contracting, employment of technology for further standardization of certain classes of contracts in order to reduce the transaction costs and time of contracting as 'electronic contracting'. In this paper we develop a logical formalism to represent the content of business contracts. This formalism can be used to develop applications that can automatically negotiate and process contracts, or it can be used to develop on-line help systems that explain to the human negotiator what for him the implications are of a certain contract that is proposed to him by the counter party. Currently a number of these on-line support systems for electronic contracting are under development [15][3]. Our formalism could be useful to further develop these on-line contract support systems.

Over the last couple of years several researchers have worked on a Formal Language for Business Communication (FLBC). This work has mainly been inspired by the work of Kimbrough and Lee. The main objective of this work is to improve the electronic communication between organizations. For example, in [8][9][14] Kimbrough and Moore convincingly argued that the traditional Electronic Data Interchange (EDI) languages such as ANSI X12 and EDIFACT have significant shortcomings. Also, the more recent developments in the field of electronic communication such as XML based messaging standards suffers from the same shortcomings identified by Kimbrough and Moore. The improvements that FLBC aims to provide can be broadly classified as expressive flexibility and clearly defined semantics. In order to achieve the latter goal Kimbrough has proposed the use of event semantics [8].

In general, FLBC is a promising candidate for enhanced electronic communication. In this paper, however, we want to explore whether FLBC can be used for another purpose, namely modeling of and reasoning about commercial agreements. The fact that FLBC intends to provide a flexible and unambiguous

communication language, has already resulted into several attempts to use FLBC as a language and protocol for negotiation [22] (see also [5]). The typical end result of a negotiation between organizations is an agreement in one form or another. In our opinion it is thus worth exploring whether the same FLBC language can be used to capture the contents of the agreement.

Since FLBC was originally intended to enhance and replace EDI messages it should be clear that it could offer solutions for capturing the data communicated. As the event semantics allow us to model events and the order in which these events occurred or should occur it seems reasonable to assume that procedures can be modeled with it. FLBC is based on the speech act theory developed by Austin, Searle and others [1], [18] and [5]. FLBC has already been used to model and structure the communication that occurs during the negotiation of an agreement [22], the communication perspective. In his FLBC work Kimbrough uses concepts such as 'kept', 'held' and 'veridical' to capture the meaning speech acts with illocutionary forces such as promise and assert. Preliminary work has already been done to link the concept of 'kept' to the concepts such as 'obligation', 'violation' and 'fulfilled' that are typically used in the deontic or normative perspective [10][4].

2. Overview of FLBC and Event Semantics

The main premise underlying FLBC is that messages exchanged between parties are made up of statements that can be analysed and represented in the F(P) framework of speech act theory [4]. Speech act theory is concerned with the analysis and representation of utterances. Speech act theory seeks to identify the main types of utterances in terms of their illocutionary aspect and to provide a systematic explication for each of these types in terms of the information conveyed between a speaker and a hearer and the conditions in which it is used successfully. Illocutionary forces are identified by verbs, which may be explicitly used by a speaker or alluded to in a given context. Hence, there are as many ways for a speaker to express the same attitude towards a given content as there are available appropriate verbs. For example, to issue a promise to meet his interlocutor a speaker might say "I promise to meet you", or "I will meet you", and so on. As there exist many verbs there are in essence an infinite number of illocutionary forces, but these fall into a small number of types, which Searle calls the illocutionary points. In his view there are five (see [18]):

1. The assertive point: used to say how the world is; used to make statements
2. The commissive point: used to commit the speaker to an action; used to make promises
3. The directive point: used to commit the hearer to an action; used to give orders
4. The declarative point: used to make changes in virtue of speaking; here, "saying so makes it so," as in an umpire crying "You're out!"
5. The expressive point: used to express the speaker's attitude; as in "Boo!".

As we aim to model the contents of agreements in this paper, we focus on speech acts with the commissive and declarative points in the remainder of this paper.

In *Events in the Semantics of English* [17] Parsons develops the view that the logical forms of simple English sentences typically contain quantification over events or states. In his work on FLBC Kimbrough introduced the idea that if event semantics is expressive enough to capture the complexities of natural languages, then it should also be expressive enough to capture the meaning of EDI messages. An essential feature of most agreements is of course that one party promises to perform certain actions, e.g. supply products, in return for certain actions by the other party, e.g. pay an amount of money. Whenever one party promises something to another party we model this as a promising event in event semantics. Typical EDI messages containing promising event are the Quotation, the Purchase Order and the Purchase Order Acknowledge documents. Kimbrough proposed to use a Kept predicate to model the propositional content of such EDI messages. For example, if one party promises (by sending an order acknowledgment EDI message) to deliver some goods to the other party, then that is modeled as:

$$\exists e\ (promise(e) \wedge speaker(e, Party) \wedge hearer(e, TheOtherParty) \wedge (kept(e) \leftrightarrow \exists e'\ (deliver(e') \wedge agent(e', Party) \wedge theme(e', Products) \wedge sake(e', e))$$

This expression is read as "There exists a promising event (speech act) with the speaker being Party and the hearer being TheOtherParty and the promising event is kept if and only if there exists another event, which is a delivery event, the agent of that delivery event is Party, what was delivered is Products and the delivery event occurred for the sake of the promising event".

Note that the Sake predicate is important to link the delivery event back to the promise that was made. Otherwise, any delivery would render the promise kept. Also, notice that one has to introduce the predicate Deliver and that other predicates, such as pay, insure, and obtain will have to be introduced in order to model other promises. Kimbrough, however, argues and claims that the domain specific vocabularies that contain such predicates can be kept to a relatively small size.

In the next sections we show how event semantics, as extended by Kimbrough, can be used to represent a model business contract drafted by the

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

International Chamber of Commerce. In the process we will highlight several areas where the event semantics as proposed by Kimbrough could be improved.

The power of Kimbrough's proposal is that with only a few constructs, such as the promises made and rights granted in the agreement, we can model a significant number of clauses that make up commercial agreements (and thus EDI messages). In this section we discuss how we could model several key constructs in event semantics.

In the previous section we introduced the representation of promising events using the *kept* predicate. Kimbrough also proposes to use the general structure of that representation for other speech acts, such as assertive speech acts. For such speech acts the *kept* predicate is replaced by other predicates such as *veridical* and *honored* (see [8] for details). It is important to note though that in his work Kimbrough uses the *kept* predicate just to represent what the promise was all about. Kimbrough does not assign any moral or legal meaning to the *kept* predicate. In other words, it is not implied that it is morally good to keep one's promises or that the speaker is legally bound to keep the promise. As the domain of application in this paper is international trade it makes sense to give the *kept* predicate the legal interpretation that "if an agent promises something, then he becomes legally bound to keep the promise". It is important to note Article 2.107 of "THE PRINCIPLES OF EUROPEAN CONTRACT LAW – 1998" in this respect [6].

Article 2.107 (ex art. 5.108)- Promises binding without acceptance

A promise which is intended to be legally binding without acceptance is binding.

We assume in this paper that all speakers of promising events have the intention that those promises are legally binding and that no acceptance is required to make the promise binding.

The representation above assumes that the promise is only kept if the party who utters the promise (the speaker) is also the agent of the delivery event. In our earlier work [19] we showed that subcontracting and thus the representation of the transfer of obligations and liabilities are critical to modeling international trade agreements. An important principle of subcontracting is laid down in Article 8.107 of the "THE PRINCIPLES OF EUROPEAN CONTRACT LAW – 1998" [6].

Article 8.107 (ex art. 3.107)– Performance Entrusted to Another

A party who entrusts performance of the contract to another person remains responsible for performance.

This principle implies that we have to distinguish between the 'transfer of agency' and the 'transfer of liability'. Subcontracting to another agent is not a means to keep one's promises in itself. The promise is only kept when the other agent actually makes the promised event happen. If that does not happen, then the original promise is not kept and the agent is liable of any consequences that might have.

We have to refine Kimbrough's proposal in order to allow for keeping a promise through actions by a third party. In [20] we presented two basic methods for doing this. For example, if we assume that a transporter will actually deliver the Products on behalf of the supplier, then we could represent this as:

$\exists e$ (promise(e) $\wedge$ speaker(e, Supplier) $\wedge$ hearer(e, Buyer) $\wedge$ (kept(e) $\leftrightarrow$
$\exists e'$ (deliver(e') $\wedge$ agent(e', Transporter) $\wedge$ theme(e', Products) $\wedge$ sake(e', e) $\wedge$ influenced(Transporter, Supplier)

In this first method the contract between the Supplier and the Transporter is 'hidden' in the *influenced* predicated. The influenced predicate is required to ensure that it was the Supplier who asked the Transporter to perform the delivery. This is stronger than just the *sake* predicate.

The second method is to make the subcontract between the Supplier and the Transporter explicit in the representation. This subcontract can be modeled as an additional promising event with its own *Kept* predicate.

$\exists e$ (promise(e) $\wedge$ speaker(e, Supplier) $\wedge$ hearer(e, Buyer) $\wedge$ (kept(e) $\leftrightarrow$
$\exists e'$ (deliver(e') $\wedge$ agent(e', Transporter) $\wedge$ theme(e', Products) $\wedge$ sake(e', e) $\wedge$
$\exists e''$ (promise(e'') $\wedge$ speaker(e'', Transporter) $\wedge$ hearer(e'', Supplier) $\wedge$ (kept(e'') $\rightarrow$
$\exists e'''$ (deliver(e''') $\wedge$ agent(e''', Transporter) $\wedge$ theme(e''', Products) $\wedge$ sake(e''', e'') $\wedge$ sake(e'', e)))

In order to link that eventual delivery event to the original promise by the Supplier we include the sake(e''',e) predicate. The problem with this is that in the real world a transportation contract between the Supplier and Transporter would not indicate that this subcontract was made in order to keep an earlier promise made by the Supplier. This would require that the Buyer somehow indicate that the actual delivery event (performed by the Transporter) *matches* the delivery event that was promised by the Supplier. Distinguishing between events that *actually* happened and events that *could* happen, however, is a general problem with the event semantics as proposed by Kimbrough. In [4] Daskalopulu and Sergot present a thorough treatment of this issue. Their suggested solution comprises the introduction of new predicates mode(e, actual) and actually_kept(e) (distinguished from

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

kept(e)) and the definition of a predicate matches, which is true in the case that at least the stipulated attributes of the promised event are all features of the actual event. We agree that this solution is a significant improvement over Kimbrough's original proposal. However, for reasons of simplicity and clarity we will ignore this issue in this paper and continue to use Kimbrough's version throughout the remainder of this paper.

Another concept that frequently occurs in agreements is conditional promises or obligations. The textbook example is a post-payment agreement, where the buyer will pay the supplier at a certain time, e.g. 30 days, after the delivery of the goods. The promise to pay is thus conditional on the delivery event. We can model this in event semantics by modeling both the delivery and the payment event.

The time when an event occurs becomes important for the representation. In line with Kimbrough's work we use the predicate cul to model when an event culminated. In a post-payment situation the delivery event has to happen before the payment event. In situations where no specific order is specified, we will assume that events are to happen simultaneously (concurrently), conform Article 7.104 of of the "THE PRINCIPLES OF EUROPEAN CONTRACT LAW – 1998" [6][1].

Article 7.104 - Order of performance
To the extent that the performances of the parties can be rendered simultaneously, the parties are bound to render them simultaneously unless the circumstances indicate otherwise.

The example below represents a post-payment agreement by stating that if the delivery event culminates at time t_1, then in order to keep the promise the payment event has to culminate at a point in time t_2 that is before the time point t_1+30 (assuming that the time points are measured in days).

$$\exists e\ (promise(e) \wedge speaker(e, Buyer) \wedge hearer(e, Supplier) \wedge (kept(e) \leftrightarrow$$
$$\exists e'\ (deliver(e') \wedge agent(e', Supplier) \wedge theme(e', Products) \wedge sake(e', e) \wedge cul(e', t_1))$$
$$\wedge$$
$$\exists e''\ (pay(e'') \wedge agent(e'', Buyer) \wedge theme(e'', Money) \wedge sake(e'', e) \wedge cul(e'', t_2) \wedge t_2 < (t_1 + 30)))$$

It is questionable though whether the above representation captures the true meaning of a conditional promise. The representation merely states that the promise is kept if two events actually happen in a certain order. This is different from the reading that "if event e' actually happened, then event e'' *should* occur afterwards in order to keep the promise". On the other hand the representation is not much different from what you find in the Interprocs Documentary Petri Net models of Lee [12][13]. For example, if event e' happens then a transition to a new state occurs. In this new state there are two possible transitions. Event e'' happens and a transition to a 'promise kept' state would occur or event e''' happens and a transition to a 'violation' state occurs. Event e''' could for example be triggered by a timer (if the 30 days deadline expires). This requires of course that you view Documentary Petri Nets as representing what procedure the parties *should* follow in order to accomplish a bureaucratic task, which differs significantly from using Petri Nets to simulate what *could* happen. If we use material implication instead of the $\wedge$ operator in order to capture the conditional aspect better we get the following representation:

$$\exists e\ (promise(e) \wedge speaker(e, Buyer) \wedge hearer(e, Supplier) \wedge (kept(e) \leftrightarrow$$
$$\exists e'\ (deliver(e') \wedge agent(e', Supplier) \wedge theme(e', Products) \wedge sake(e', e) \wedge cul(e', t_1))$$
$$\rightarrow$$
$$\exists e''\ (pay(e'') \wedge agent(e'', Buyer) \wedge theme(e'', Money) \wedge sake(e'', e) \wedge cul(e'', t_2) \wedge t_2 < (t_1 + 30)))$$

The problem with using material implication is that it has the property that the implication is true when the antecedent is false. In other words, if event e' did not happen, then the promise would be kept, which is counterintuitive. In this paper we do not further explore whether other conditional operators than material implication could improve the representation. Instead we draw attention to the fact that what we just described as counterintuitive is actually a more general problem with the Kept predicate. The Kept predicate as introduced by Kimbrough intends to capture the meaning of a promise by specifying when it is kept or not kept. The distinction between kept and not kept, however, is not fine grained enough. A promise (or obligation) can be in four different states, which we will call (following the work of Lee [11]) 'not-triggered', 'pending', 'kept' (fulfilled) and 'not kept' (violated). The 'not-triggered' state is only applicable to conditional promises (obligations) and represents the situation in which the condition is not true. The 'pending' state represents the situation in which a promise has not been kept yet, but it is still possible to keep the promise. The 'not-kept' state represents the situation in which the promise was not kept and it is no longer possible to keep the promise. In the next section we have a closer look at the situations in which a promise has not been kept.

[1] Note that this is similar to the principle used by Ron Lee for generating Documentary Petri Nets based on Procedure Constraint Grammars. Lee states that "Actions are considered to be temporally unordered (concurrent), unless specified in constraints" [page 6, L98].

In the examples above we modeled a promise by specifying what events should occur in order to keep the promise. By inspecting the events that actually occurred we could infer whether or not a promise has been kept. If the promise is not kept, then that will typically have certain consequences. A breach of contract, which is a breaking of a duty or obligation that a contract imposes, confers a right of action for damages on the injured party [20]. It also entitles the injured party to treat the contract as discharged if the other party is in fundamental breach, that is, repudiates the contract. Hence, not keeping a promise should be classified as a 'fundamental breach' or a 'non-fundamental breach', because of the difference in the remedies available to the injured party. The contract law of the European Union set out in "THE PRINCIPLES OF EUROPEAN CONTRACT LAW – 1998" contains the following definition of a fundamental non-performance of an obligation:

Article 8.103 - Fundamental Non-Performance
A non-performance of an obligation is fundamental to the contract if:

a. strict compliance with the obligation is of the essence of the contract; or
b. the non-performance substantially deprives the aggrieved party of what it was entitled to expect under the contract, unless the other party did not foresee and could not reasonably have foreseen that result; or
c. the non-performance is intentional and gives the aggrieved party reason to believe that it cannot rely on the other party's future performance.

It is advisable to include more explicit rules regarding what constitutes a fundamental breach of contract in an agreement. The ICC model *Distributorship Contract* uses this approach as can be seen from article 19.3 of the model contract [7]. The first part of Article 19.3 reads:

"The parties hereby agree that the violation of the provisions under [list of article numbers] of the present contract is to be considered prima facie evidence of a substantial breach of the contract."

In order to distinguish the provisions (promises) whose breach would be considered fundamental (or substantial) from the other provisions we introduce two new predicates, kept_fundamental and kept_nonfundamental. As it seems reasonable to assume that the keeping of the promise to delivery the products is fundamental to contracts for sale of goods, we would model this promise as:

$\exists e$ (promise(e) $\wedge$ speaker(e, Party) $\wedge$ hearer(e, TheOtherParty) $\wedge$ (kept_fundamental(e) $\leftrightarrow$
$\exists e'$ (deliver(e') $\wedge$ agent(e', Party) $\wedge$ theme(e', Products) $\wedge$ sake(e', e))

The following articles from the *Convention on the International Sale of Goods* (CISG) illustrate that a non-fundamental breach can have very different consequences than a fundamental breach [21].

Article 48
(1) *Subject to article 49, the seller may, even after the date for delivery, remedy at his own expense any failure to perform his obligations, if he can do so without unreasonable delay and without causing the buyer unreasonable inconvenience or uncertainty of reimbursement by the seller of expenses advanced by the buyer. However, the buyer retains any right to claim damages as provided for in this Convention.*
[subclauses (2), (3), (4) omitted]

Article 49
(1) The buyer may declare the contract avoided:
(a) if the failure by the seller to perform any of his obligations under the contract or this Convention amounts to a fundamental breach of contract
(2) However, in cases where the seller has delivered the goods, the buyer loses the right to declare the contract avoided unless he does so:
(a) *in respect of late delivery, within a reasonable time after he has become aware that delivery has been made;*
[subclauses 1(b) and 2(b) omitted]

Article 48 stipulates that if the breach is non-fundamental, then the seller might have the right to remedy the breach. After remedying the breach the buyer will still be bound to his obligations. If the breach was fundamental, then the buyer may declare the contract avoided. The supplier has no right to remedy the breach and after avoiding the contract the buyer is no longer bound to perform his obligations, such as pay for the goods.

As a fundamental breach can have such far reaching consequences, the CISG makes clear that not every incorrect delivery is automatically a fundamental breach as can be seen from Article 49 (2). In others words, it seems that you can keep your promise 'to a certain degree' and that the degree to which you kept your promise has an influence on the consequences.

In the previous section we touched upon the issue of not keeping a promise and the consequences that might have. We discussed that not keeping a promise is certainly not always a fundamental breach of contract that could result in the termination of the contract. In many cases a situation in which a promise has not been kept can

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

easily be remedied. For example, if a payment event did not occur within the period of 30 days after delivery, as was agreed in the contract, then the buyer could still make the payment, possibly with an interest charge added to the price. Such contingencies can often be foreseen and are typically catered for in the agreement by means of (explicitly described) remedies. For example, we could say that the buyer could either keep his promise by paying within 30 days or by paying the contract price increased with a 1% penalty after 30 days. We can model this as:

$\exists e$ (promise(e) $\wedge$ speaker(e, Buyer) $\wedge$ hearer(e, Supplier) $\wedge$
(kept_fundamental(e) $\leftrightarrow$
$\exists e'$ (deliver(e') $\wedge$ agent(e', Supplier) $\wedge$ theme(e', Products)
$\wedge$ sake(e', e) $\wedge$ cul(e', t_1))
$\wedge$
[($\exists e''$ (pay(e'') $\wedge$ agent(e'', Buyer) $\wedge$ theme(e'', Money) $\wedge$
sake(e'', e) $\wedge$ cul(e'', t_2)) $\wedge$ $t_2 < (t_1 + 30)$))
$\vee$
($\exists e''$ (pay(e'') $\wedge$ agent(e'', Buyer) $\wedge$ theme(e'',
MoneyPlusPenalty) $\wedge$ sake(e'', e) $\wedge$ cul(e'', $t3$) $t3 > (t_1 +$
30)))]

The problem with this representation is that in some situations an agent might want to use the representation to plan his action beforehand in such a way that he complies with the rule that keeping your promise by paying within 30 days is actually the preferred way of keeping your promise. In order to accomplish this we could introduce two kept predicates, which can be used to specify a preference ordering between those kept predicates. For example, we could refine the representation as follows, where high and low are indications of the preference:

$\exists e$ (promise(e) $\wedge$ speaker(e, Buyer) $\wedge$ hearer(e, Supplier) $\wedge$
[(kept_fundamental(e, high) $\leftrightarrow$
($\exists e'$ (deliver(e') $\wedge$ agent(e', Supplier) $\wedge$ theme(e', Products)
$\wedge$ sake(e', e) $\wedge$ cul(e', t_1))
$\wedge$
$\exists e''$ (pay(e'') $\wedge$ agent(e'', Buyer) $\wedge$ theme(e'', Money) $\wedge$
sake(e'', e) $\wedge$ cul(e'', t_2)) $\wedge$ $t_2 < (t_1 + 30)$))))
$\vee$
(kept_fundamental(e, low) $\leftrightarrow$
($\exists e'''$ (deliver(e''') $\wedge$ agent(e''', Supplier) $\wedge$ theme(e''',
Products) $\wedge$ sake(e''', e) $\wedge$ cul(e''', t_3)) $\wedge$ t
$\wedge$
$\exists e''''$ (pay(e'''') $\wedge$ agent(e'''', Buyer) $\wedge$ theme(e'''',
MoneyPlusPenalty) $\wedge$ sake(e'''', e) $\wedge$ cul(e'', t_4)) $\wedge$ $t_4 > (t_3 +$
30)))))]

Using the kept_fundamental(e,high) and kept_fundamental(e,low) we can differentiate between the different ways in which a promise was kept. It is important to note that this distinction can be used for two very different purposes, planning and monitoring. By *planning* we mean that an agent would use the contract representation to optimize her behavior in the planning stage before the action is executed. For example, if the objective of the agent is to behave in the best possible manner from a deontic point of view, then the agent could try to make those events happen that lead to the maximum number of kept_fundamental(e,high). A more calculative agent might prefer to plan her actions in such a way that she executes the contractual promises in a minimal way at the lowest costs. Hence, she could, for example, plan to comply with the contract but with as many of her promises kept_fundamental(e,low) as possible. Note that this planning is a separate decision making process that takes as input the representation of the content of the contract, but it cannot be automatically inferred in first order logic from the event semantics representation of the content of a contract. The representation enables this kind of decision making by providing syntactic cues like 'high' and 'low', but the decision making itself is left to the agent. In the case of an artificial intelligent agent this decision making process could be executed by a separate expert system inside the agent for planning the actions of the agent. However, this expert system is beyond the scope of what we are modeling here, namely the content of the contract. By *monitoring* we mean to monitor to what extent the behavior of the agent complies with the contract. No attempt is made to help the agents to decide what to do. Monitoring is typically an ex-post activity, i.e. executed after the actions of the agent took place, whereas planning is typically an ex-ante activity, which is done before the action is executed. During monitoring we analyze what has actually happened. Instead of using the preference ordering to optimize behavior, it is only observed whether promises where kept and in which way (fundamental or not, high or low etc.) One could compare the monitoring function with the perspective of a judge in a court case. The judge decides whether a person violated the law, but he cannot make the person act in such a way that she complies with the law.

This fundamental distinction between planning and monitoring is related to the distinction introduced in [10] between representing the meaning of a promise in an EDI message in event semantics, which was called *meaning unfolding*, and representing that keeping a promise was obligatory, which was called *unwrapping*. The main point is that making a promise (for the other) and the commitment to keep a promise (for yourself) are two fundamentally different things. In [10] the focus is on modeling EDI messages rather than full contracts. One could argue that the distinction between meaning unfolding and unwrapping at EDI level is analogous to the distinction between monitoring and planning at the level of full contracts.

In the previous sections we have discussed in detail how we can model promising events in event semantics. In this section we look at events where the

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

speaker grants a right to the hearer. For example, in Article 1.1 of the ICC model contract it is stipulated that "The Supplier grants and the Distributor accepts the exclusive right to market and sell the Products in the Territory." This can be modeled in event semantics as follows:

∃e (grant(e) ∧ speaker(e, Supplier) ∧ hearer(e, Distributor) ∧ (kept(e) ↔
¬ ∃e'∃e' ' (market(e' , Products) agent(e' , Distributor) ∧ failure(e')∧ agent(e' ' , Supplier) cause_failure(e' ' , e'))

(With a slight abuse of notation we use failure(e') both as formula and as term in this case.) This expression is read as "There exists a granting event, with speaker Supplier and hearer Distributor and the granting event is kept if in case of a not-successful attempt to market and sell the products by the distributor, then this was not caused by an action of the Supplier". In other words, if you grant a right, then you are not allowed to prevent (interfere with) the grantee excersing his right.

Note that this reading is quite different from the standard deontic interpretation of permission. In standard deontic logic, permission is the dual of obligation in the sense that an action is permitted if and only if it is not obligatory that this action is not performed.[2] This standard deontic interpretation has some disadvantages. Firstly, it follows from the definition that everything that is not explicitly forbidden is permitted.[3] This is at odds with several clauses of the model contract. In these clauses the supplier specifically grants rights to the distributor. Certainly not all rights granted in the contract were explicitly forbidden before the granting event. Some, such as the use of the supplier's trademark are explicitly forbidden in international trademark law. However, where is it explicitly stated that you, are not allowed to (re)sell the supplier's products in a certain territory without the supplier's permission? Secondly, it is defined from the point of view of the grantee and not the grantor. As we use speech acts as the basic construct it seems to be more natural to define the granting of a right (giving a permission) in terms of the grantor (speaker). See [20] for more details on the definition of directed permission.

3. Using Prolog to implement Event Semantic

In Section 2 we have elaborated how several key constructs of agreements can be modeled in event semantics. In the introduction we mentioned that there is a wide variety of initiatives to support contract drafting, negotiation and drafting on the Internet. In order to enhance these initiatives with the kind of inferences that the representation of contracts in event semantics would enable, an inference engine is required. In this section we, therefore, give an indication of how Prolog could be used to implement and reason about the event semantic representation introduced in section 2. For a detailed treatment of how event semantics can be implemented using Prolog we refer the reader to Daskalopulu and Sergot [4]. In this paper we only aim to give the reader an impression of how some of the key constructs could be implemented in Prolog. We start again with the simple promising event:

∃e (Promise(e) ∧ Speaker(e, Party) ∧ Hearer(e, TheOtherParty) ∧ (kept(e) ↔
∃e' (Deliver(e') ∧ Agent(e' , Party)∧ Theme(e', Products) ∧ Sake(e' , e))

Even though we said in section 2 that we would ignore the distinction between events that *actually* happened and events that *could (or should)* happen, we have to make at least an implicit distinction in Prolog. The promising event has actually happened and is thus represented as facts in Prolog.

```
Promise(e).
Speaker(e, ' Party' ).
Hearer(e, ' TheOtherParty' ).
```

The propositional content of the promise can be represented in Prolog as[4]

```
Kept(e) :- Deliver(X), Agent(X, ' Party' ), Theme(X, ' Products' ), Sake(X, e).
```

In order to infer that the promise was kept we need to have a ' matches algorithm' that would assert

```
Deliver(e1).
Agent(e1, ' Party' ).
Theme(e1, ' Products' ).
Sake(e1, e).
```

[2] Standard Deontic Logic is one of the first and also simplest deontic logic that are developed. After that a wide variety of more sophisticated logics have been developed. For more details on standard deontic logic and other types of deontic logic see, for example, [2] or [16].

[3] Assuming that ' forbidden(A)' is equivalent to ' obligation(not A)' .

[4] The uni-directional implication :- in the Prolog rule is analogous to the bi-directional implication in the event semantics rule rule Kept(e) ↔ ∃e' (…) due to the Clark Completion of the Prolog implication. The intuition behind this is that if p :- q is the only rule about p in the program, then you know in Prolog that if p is derivable, then q is also true.

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

in case an actual event matched the promised event. Let's now look at the more complicated promise where we have two alternative ways in which it can be fulfilled.

$\exists e$ (promise(e) $\wedge$ speaker(e, Buyer) $\wedge$ hearer(e, Supplier) $\wedge$
[(kept_fundamental(e, high) $\leftrightarrow$
($\exists e'$ (deliver(e', Products) $\wedge$ agent(e', Supplier) $\wedge$ theme(e', Products)
$\wedge$ sake(e', e) $\wedge$ cul(e', t_1))
$\wedge$
$\exists e''$ (pay(e'', Buyer) $\wedge$ theme(e'', Money) $\wedge$
sake(e'', e) $\wedge$ cul(e'', t_2) $\wedge$ $t_2 < (t_1 + 30)$))))
$\vee$
(kept_fundamental(e, low) $\leftrightarrow$
($\exists e'''$ (deliver(e''', Supplier) $\wedge$ agent(e''', Supplier) $\wedge$ theme(e''', Products) $\wedge$ sake(e''', e) $\wedge$ cul(e''', t_3)) $\wedge$
$\exists e''''$ (pay(e'''', Buyer) $\wedge$ agent(e'''', Buyer) $\wedge$ theme(e'''', MoneyPlusPenalty) $\wedge$ sake(e'''', e) $\wedge$ cul(e'''', t_4), $\wedge$ $t_4 > (t_3 + 30)$))))]

This promise can be implemented in Prolog as follows:

```
Promise(e).
Speaker(e, ' Buyer' ).
Hearer(e, ' Supplier' ).

Kept_fundamental(e, ' high' ) :
Deliver(X), Agent(X, ' Supplier' ), Theme(X, Products),
Sake(X, e), Cul(X, t1), Pay(Y), Agent(Y, ' Buyer' ), Theme(Y,
' Money' ), Sake(Y, e), Cul(Y, t2), Before(t2, (t1 + 30)).

Kept_fundamental(e, ' low' ) :
Deliver(X), Agent(X, ' Supplier' ), Theme(X, Products),
Sake(X, e), Cul(X, t1), Pay(Y), Agent(Y, ' Buyer' ), Theme(Y,
' MoneyPlusPenalty' ), Sake(Y, e), Cul(Y, t2), After(t2, (t1 +
30)).
```

Note that the disjunction in the event semantic representation translates into two separate rules in Prolog.

Even though it has some limitations the implementation of the promising event is relatively straightforward. The implementation of the granting event is more complicated.

$\exists e$ (grant(e) $\wedge$ speaker(e, Supplier) $\wedge$ hearer(e, Distributor)
$\wedge$ (kept(e) $\leftrightarrow$
$\neg$ $\exists e'$ $\exists e''$ (market(e', Products) $\wedge$ agent(e', Distributor) $\wedge$
failure(e') $\wedge$ agent(e'', Supplier) $\wedge$ cause_failure(e'', e'))

In order to implement this we need a rule to identify that there was either no market for the distributor or no market failure that was caused by the supplier.

```
Grant(e).
Speaker(e, ' Supplier' ).
Hearer(e, ' Distributor' ).

Kept(e) :- not Market_failure(X, ' Distributor' )
Kept(e) :- not Cause_failure(X, ' Supplier' ).
Market_failure(X, ' Distributor' ) :- Market(X, ' Products' ),
Agent(X, ' Distributor' ), Failure(X).
```

Here 'not' is the negation-by-failure of Prolog.

4. Conclusions

In this paper we applied and further developed the Formal Language for Business Communication (FLBC) of Kimbrough, which is based on Parsons' event semantics. Kimbrough mainly developed FLBC to model simple EDI-based messages in business communication. We investigated if and how FLBC could be applied to model full business model contracts. As an example of a contract we applied FLBC to various model contracts from the International Chamber of Commerce. Our analysis has shown that in principle the key elements of business contracts can indeed be modeled using FLBC. However, we also argued that FLBC needs to be enhanced in several ways to do the actual modeling of contracts. Based on this analysis we gave several suggestions for a possible further development of FLBC. We also showed how specific FLBC constructs for modeling contracts could be implemented in Prolog.

5. References

[1] Austin, J. L. *How To Do Things With Words*, 2nd ed. Harvard University Press, 1975.

[2] Chellas, B., *Modal Logic: an introduction*, Cambridge University Press, 1980.

[3] http://www.dicarta.com/

[4] Daskalopulu, A., and Sergot, M., "Computational Aspects of the FLBC Framework", Decision Support Systems, to appear, 2001.

[5] Dignum F. and Weigand H. Communication and deontic logic. *Information Systems, Correctness and Reusability*, *World Scientific, Singapore* (1995) pp. 242-260.

[6] European Union, "THE PRINCIPLES OF EUROPEAN CONTRACT LAW – 1998", 1998, http://www.jus.uio.no/lm/eu.contract.principles.1998/doc.html

[7] International Chamber of Commerce , "The ICC Model Distributorship Contract", ICC Publication N518, ICC Publishing, Paris, 1993

[8] Kimbrough, S., "Formal Language for Business Communication (FLBC): Sketch of a basic theory", *International Journal of Electronic Commerce*, Vol. 3, 2 (Winter 1998 – 99), pp. 23-44.

[9] Kimbrough, S. O., and Moore, S. A. "On automated message processing in electronic commerce and work support systems:

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

Speech act theory and expressive felicity" *ACM Transactions on Information Systems 15*, 4 (October 1997), 321-367.

[10] Kimbrough, S.O, and Tan, Y.H., "On Lean Messaging with Wrapping and Unfolding for E-Commerce", *International Journal of Electronic Commerce*, Vol. 5, Nr. 1, 2000. pp. 83-108.

[11] Lee, R.M., "A Logic Model for Electronic Contracting", *Decision support systems*, Vol. 4, 1988, pp. 27-44.

[12] Lee, R.M., "Towards Open Electronic Contracting", *International Journal of Electronic Markets Special Issue on Electronic Contracting (ed. S. Micossi)*, Vol 8, No 3, 1998, pp. 3-8.

[13] Lee, R.M., "Distributed Electronic Trade Scenarios: Representation, Design, Prototyping", International Journal of Electronic Commerce, Vol. 3, No. 2, February 1999.

[14] Moore, S.A., "Categorizing automated messages", *Decision Support Systems*, 22, 3, pp. 213-241

[15] MEdiating and MOnitoring Electronic Commerce, ESPRIT Project 26.895, http://www.abnamro.com/memo

[16] Meyer, J-J.Ch., and Wieringa, R. (Eds). *Deontic Logic in computer science*. Chichester, Wiley, 1993.

[17] Parsons, T., *Events in the Semantics of English: A Study in Subatomic Semantics*, The MIT Press, 1990.

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

[19] Tan, Y.H., and Thoen, W., "A logical model of transfer of obligations in trade contracts", *Accounting, Management and Information Technologies*, Pergamon, Vol 8., No. 1., pp. 23-38, 1998.

[20] Tan, Y.H. and Thoen, W., "A Logical Model of Directed Obligations and Permissions to Suppport Electronic Contracting", *International Journal of Electronic Commerce (IJEC)*, Vol 3. No.2, pp. 87-104, winter 1998-99.

[21] United Nations, Convention On Contracts For The International Sale Of Goods, Vienna, 1980, http://www.jus.uio.no/lm/un.contracts.international.sale.of.goods.convention.1980/index.html

[22] Weigand H. and Heuvel W-J., "Meta-Patterns for Electronic Commerce Transactions Based on the Formal Language for Business Communication (FLBC)", *International Journal of Electronic Commerce*, Vol. 3, 2, (winter 1998-99), pp. 45 – 65.

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE