Representing and Enforcing Electronic Commerce Contracts Over A Wide Range of Platforms Using Occurrence Stores

Alan S. Abrahams and Jean M. Bacon
University of Cambridge Computer Laboratory
{asa28, jmb}@cam.ac.uk

1 Introduction

We present a novel approach for representing, storing, and enforcing electronic commerce contracts over diverse platforms using occurrence stores. We first define what is meant by an occurrence and demonstrate how an occurrence may be used to store a workflow event, such as a purchase. A discussion of contracts and their representational requirements follows, and we show how occurrences may be used to represent contractual provisions (policy) such as defined terminology, regular and conditional duties, permissions, and powers in contracts. We show that the storage of queries is necessary to allow us to store contractual provisions, because stored queries allow us to determine which stored descriptions describe a given concept or occurrence, and consequently which policies – contractual provisions – apply to that concept or occurrence. We provide an overview of our implementation, which is capable of storing and interpreting commercial contracts over heterogeneous platforms. Scalability and fault tolerance may be achieved by replicating stored policies to multiple enforcement servers.

2 Representing and Storing Occurrences

An occurrence is a happening that occurs at a particular point in time or that occupies an interval of time. Occurrences have participants acting in various roles. For example, a purchasing occurrence has participants acting in the roles `purchaser`, `purchased`, and `seller`. The occurrence "Brian's selling of his book" can be considered as "an entity (`c1`) named 'Brian' participates as `seller` in an occurrence (`o1`), classified by some authority as a sale/purchase, with an entity (`c2`), classified by some individual or institution as a book, participating as the `sold` item". We have implemented a formal system inspired in part by Davidsonian event semantics to represent and reason about commercial contracts and occurrences in a platform independent manner. Occurrences may be represented in a tabular form, which defines the occurrence identifier, the participants in the occurrence, and their respective roles. Our motivation for choosing this particular participant-occurrence-role implementation schema is that it provides a simple, convenient, and uniform means of storing and retrieving occurrences, queries and policies (contractual provisions) across heterogeneous relational database platforms. *Table 1* below shows the tabular representation of the occurrence `o1` which is a purchasing of a particular book by Walt, from Brian.

Participant	Occurrence	Role
c1 (named 'Brian')	o1	seller
c2 (classified as a book)	o1	bought
c3 (named 'Walt')	o1	buyer

Table 1: Representing a purchasing occurrence "Walt buying the book from Brian"

Rather than store a particular identified participant, we may specify that the participants are all concepts that satisfy a query or that are in the range of a mapping function. In this case, rather than store each concept-identifier that satisfies the query, we store a reference to the query or mapping function, such as the query reference ?q3. ?q3 has the semantics that all results of that query (q3) are participants in the occurrence. The next section discusses the storage of contracts and the relevance of query storage to contractual provision (policy) applicability determination.

3 Representing and Storing Contracts

In order to represent commercial contracts, formalisms must be provided to represent contractual provisions – that is, the descriptive and prescriptive policies specified in the contract. The provisions of a contract include policies that define terminology and its interpretation, regular and conditional rights and duties of the parties, and authority (permissions and powers). We must be able to specify what constitutes a breach of contract. The formalisms needed to represent these requirements in an occurrence centric manner are explained in the following sections.

3.1 Terminology and the Interpretation of Terms

Terminology is defined so as to explain how the parties name and classify items and occurrences being referred to by the contract. Defined terminology can be easily implemented using a combination of classifying and naming occurrences. *Table 2* below defines the term 'overdue' according to the definition 'accounts with balance > $100 for longer than 60 days are classified as overdue' specified in a contract. Classifying occurrences take the form '[individual, institution, contract, or contractual provision, the `classifier`] classifying [concept or set of concepts, the `classified` item(s)] as [concept, the `class`].' The class is then usually named using a naming occurrence.

Participant	Occurrence	Role
?q1 (the results of the query/description 'accounts with balance > $100 for longer than 60 days')	o4 (a classifying)	classified concepts
c2 (the concept *overdue*)	o4	class
c4 (the contract making the classification)	o4	classifier
c2 (the concept *overdue*)	o5 (a naming)	named
s8 (the symbol 'overdue')	o5	name

Table 2: Representing a classifying occurrence "Accounts with balance > $100 for longer than 60 days are classified as overdue"

The interpretation of a term is the results obtained by resolving the query associated with the term. In the case of the term 'overdue' defined in *Table 2* above, 'overdue' is the `name` of the `class c2` associated with the classifying occurrence `o4`. The interpretation of the class `c2`, which is named 'overdue', would be `?q1` – that is the results of the query `q1` – which holds the role `classified` in the classifying occurrence `o4`. The inverse of query resolution – resolving a query – is query coverage determination – i.e. determining what stored queries cover a given item. To find what a particular account is regarded as, we determine analytically what queries cover the account. For instance, we could determine that an account with a balance of $125 for 72 days would be interpreted as 'overdue' as it is covered by the query 'accounts with balance > $100 for longer than 60 days' which defines the set of items classified as overdue, whereas an account with a balance of $20 for 30 days would not be regarded as overdue.

The use of classifying occurrences allows us to construct ontologies in a more flexible manner than is possible with object-oriented languages. Object-oriented languages expect the static class hierarchy to be created first – objects are instantiated according to fixed class definitions; this inhibits evolution and precludes multiple interpretations. Our approach allows alternative classifications and retrospective reclassifications, through the simple mechanism of adding classifying occurrences. Furthermore, unlike object-oriented class structures, classifying occurrences are subjective, allowing for multiple classifications by different parties or by the same party over time.

3.2 Duties of Parties

A duty or obligation can be represented in the form: [obliger] obliges [occurrences fitting a description]. *Table 3* below represents the duty of Brian to deliver a particular book to Walt by 26 May 2001. This duty is equivalent to the entitlement of Walt to have the particular book delivered to him.

Participant	Occurrence	Role
c3 (named 'Walt')	o4 (an obliging)	obliger
?q1 (the results of the query/description 'a single occurrence of delivering of the particular book by Brian to Walt by 26 May 2001')	o4	obliged occurrences

Table 3: Representing the duty (obligation `o4`) of Brian to deliver the book to Walt

Storing the semantics of obligations allows the system to determine what occurrences were obliged, and consequently whether a given occurrence which it is capable of triggering may qualify as satisfying an obligation. Violations of obligations are discussed in *Section 3.5: Breach of Contract: Violations*.

3.3 Conditional Duties

It is common for duties in contracts to be **conditional obligations** that come into force upon occurrence of a certain event. For instance, the duty of Brian to deliver is contingent upon Walt's paying for the book. The conditional nature of the obligation can be achieved by classifying the occurrence `o4` as an 'obliging' (i.e. a valid and in-force obligation according to a certain party or institution) **only if** the conditions have been met. This is implemented using a classification of the results of a conditional query, as illustrated in *Table 4* below; the conditional query returns the occurrence `o4` only if the conditions have been met, and consequently `o4` is classified as an in-force obligation only if the conditions have been met. In the absence of being classified as an 'obliging' ('in-force obligation'), `o4` is not an enforceable obligation. The role names 'obliger' and 'obliged occurrence' (in *Table 3* above) have no semantics in the absence of an explicit classification of the occurrence

with which these roles are associated (in *Table 4* below). So the occurrence ○4 becomes an in-force obligation only when it is classified as such by a particular institution, and does ***not*** become an obligation solely by virtue of having the roles 'obliger' and 'obliged occurrences'.

Participant	Occurrence	Role
?q5 – this is a conditional query which: • when 'Walt has paid Brian' exists, returns the occurrence o4 – in *Table 3* above - which has 'Brian' as obliger and 'delivering the book to Walt by 26 May 2001' as the obliged occurrence • when 'Walt has paid Brian' does not exist, returns nothing	o6 (a classifying)	classified
c6 (an obliging / an in-force obligation)	o6	class
c7 (the contract)	o6	classifiers

Table 4: Representing a conditional obligation of Brian to deliver the book to Walt if Walt has paid for the book, using a conditional query in a classifying occurrence

3.4 Authority: Permissions and Power

'The salesperson is authorized to sell items discounted by up to 50% of their regular sales price' could mean one or both of two things:

1. 'The salesperson is **prohibited** from selling items discounted by more than 50% of their regular sales price'. The representation of this prohibition is as depicted in *Table 5* below. Consider the newly added occurrence, ○91, which is an occurrence of selling a computer regularly priced at ∈50 for ∈20. It can be determined that ○91 fits the description (that is, is covered by the query) 'occurrences of selling where goods are sold at a discount of more than 50% of their regular price' and it can therefore be concluded that the occurrence ○91 was prohibited by the stored prohibition, and that some corrective action ought to be taken. The corrective action may entail disciplinary action against the salesperson, or some other recourse.

Participant	Occurrence	Role
c1 (the company)	o5 (a prohibiting)	prohibitor
?q1 (the results of the query/description "occurrences of selling where goods are sold at a discount of more than 50% of their regular price")	o5	prohibited

Table 5: Representing a company prohibiting a salesperson from selling goods at a discount of more than 50% of their regular price

2. 'The salesperson is **empowered to** sell (**contractually capable of** selling) only items discounted by less than 50% of their regular sales price'. The **power** of an individual to bring about an occurrence means that the occurrence, provided it fits certain criteria, will be construed as (i.e. classified as) an occurrence of a certain type by a governing institution – in this case, the occurrence is only classified as a valid sale if it fits certain criteria. The representation of this power is as depicted in *Table 6* below.

Participant	Occurrence	Role
?q1 (the results of the query/description "occurrences of selling where goods are sold at a discount of less than 50% of their regular price")	o6 (a classifying)	classified
c1 (the company making this classification)	o6	classifier
c4 (a valid selling)	o6	class

Table 6: Representing the power of a salesperson to sell goods (enact a valid occurrence of selling) only when they are discounted by less than 50% of their regular price

In the example "*Salespeople* may not sell goods discounted by more than 50%" notice that '*salespeople*' is a query which returns the set of individuals classified as salespeople. If John Smith is a salesperson, the query '*salespeople*' resolves to a set of results that includes John Smith, yielding the policy "*John Smith* may not sell goods discounted by more than 50%". Determination of covering-queries – the reverse of query-resolution – may be used to check which policies apply to John Smith. A determination of covering-queries for *John Smith* would reveal that he is covered by the query '*salespeople*', and that *salespeople*, are in turn, covered by the nested query (in this case, a query which returns occurrences) '*occurrences of salespeople selling goods discounted by more than 50%*', which in turn is covered by the prohibition against such occurrences (i.e. is covered by the 'prohibiting' occurrence that takes the afore-mentioned occurrence-description in the 'prohibited' role). The determination of covering queries is therefore an essential tool in interrogating the contract in order to ascertain which provisions of the contract apply. If there are multiple conflicting provisions we have to choose which one to *apply*: that is, whether a given permission or specific obligation overrides a contradictory prohibition or vice versa.

3.5 Breach of Contract: Violations

Violations occur when an obligation to perform an occurrence fitting a description has not been fulfilled by a certain deadline or when a forbiddance (prohibition) against performing an occurrence fitting a certain description has been flouted. An obligation of Brian to deliver a particular book to Walt by 26 May 2001 is violated if there is no such delivery within the requisite time, and can be represented as in *Table 7* below. Secondary obligations (which are a form of conditional duties, which were discussed in *Section 3.3* above) may come into force when new occurrences of violations arise.

Participant	Occurrence	Role
?q7 – the results of a conditional query which: • after the deadline, and if there are no occurrences fitting the description of obliged occurrences (i.e. fitting the description in the 'obliged' role of the obligation o4[1] of Brian to deliver the book to Walt by 26 May 2001), returns the obligation o4. • before the deadline, returns nothing.	o8 (a violating)	violated
o8	o9 (a classifying)	classified
c12 (a violation/breach, in terms of the contract)	o9	class

Table 7: Representing the violation of an obligation to perform by a deadline if the deadline passes with no suitable fulfillment occurrence for that obligation

4 Performing and Enforcing Contracts

An occurrence-based system performs and enforces contractual provisions (policies) by:

- **determining currently applicable obligations**, by finding descriptions of obliged occurrences associated with occurrences currently classified as valid obligings (**diagnosis**)
- **fulfilling its own obligations** by triggering occurrences which it is capable of which fit the descriptions specified in its obligations (**liveness**)
- **avoid violating prohibitions** which cover it, by trying not to trigger occurrences which fit the descriptions specified in such prohibitions (**safety**)
- **monitoring fulfillment** of (primary and secondary) obligations and violation of prohibitions by users, through the determination of covering-queries (**detection**)
- **classifying** new occurrences appropriately according to defined norms, thereby causing certain classifications of occurrences (in line with powers) and suppressing certain classifications of occurrences (**prevention**)
- **fulfilling any secondary obligations (conditional obligations)** arising from unavoidable violations (**cure / dependability**)

Our implementation makes use of a continuous query evaluation mechanism which determines how the addition of new concept identifiers and occurrences to the occurrence store affects the results of stored queries. The continuous query evaluation mechanism determines which queries (descriptions) cover the newly added concept as well as which queries begin to cover other concepts as a result of the newly added concept. The system can then determine which policies apply to the newly added concept and which policies begin to apply to other concepts because of the newly added concept.

5 Implementation

The simple participant-occurrence-role structure of all occurrences enables us to use a broad variety of back-end data stores to uniformly store occurrences, as well as queries and policies (contractual provisions). Our policy storage environment, EDEE, has been successfully tested against Oracle 8.1.7, PostgreSQL 7.1, IBM DB/2 7.1, Microsoft SQL Server 7.0 (SQL Server 2000) and Microsoft Access 2000 running on SunOS, Red Hat Linux 6.2, Windows NT and Windows 2000. The ability to uniformly store queries and policies in a vendor-independent manner provides greater platform-independence than proprietary stored-procedures and triggers, and allows distribution of the policies across heterogeneous remote sites. Even databases that do not support stored procedures and triggers can be used for query and policy storage. The continuous query evaluation mechanism provided by EDEE improves safety by allowing the system to automatically determine applicable policies. The programmer does not have to explicitly specify all events that may cause a policy to become applicable, as is the case with traditional Event-Condition-Action triggers, where all potential triggering events must be explicitly specified. As policies (commercial contracts) are stored as data, scalability and dependability may be achieved by replicating stored policies to multiple enforcement servers. Contractual provisions can therefore be deployed in a scalable, safe, and platform-independent manner.

[1] Refer to *Table 3* on page 2 for the definition of the obligation (obliging) occurrence o4.