Mapping legal cases to RDF named graphs using a minimal deontic ontology for computer-assisted legal inference

Alan S. Abrahams
Wharton School
University of Pennsylvania
asa28@wharton.upenn.edu

David M. Eyers
University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge, United Kingdom
{firstname.lastname}@cl.cam.ac.uk

Jean Bacon
University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge, United Kingdom
{firstname.lastname}@cl.cam.ac.uk

ABSTRACT

The complexity of natural language employed within legal documents will leave their meaning opaque to automated computer interpretation for many years to come. Even so, this paper proposes that informative inference of deontic state can still be performed over a simplified encoding of legal semantics. We discuss how to map numerous deontic concepts into our minimal encoding. We use RDF named graphs to independently represent the numerous conflicting perspectives on the state of affairs being examined within a legal case. Inference over these named graphs is performed using an implementation of event calculus. We encode the key parts of a commercial case from the High Court of South Africa, and demonstrate that useful inferences can be formed on the basis of the minimal encoding.

1. INTRODUCTION

Justice and the law, internationally, are immense resource consumers in terms of staff time, money, and both electronic and physical records. Increasingly, legal information is publicly available through the World Wide Web: many countries provide summarised representations of their courts' cases.

As Semantic Web technologies mature, there are significant increases in the potential for computers to be able to correlate legal material from distributed information sources. By better organising public legal data, we believe that it will become more accessible to the public, as well as increasing the efficiency of legal practitioners.

Compared to the explosive expansion of the Web, though, the Semantic Web is growing more slowly. It is unlikely that ontologies and parsing techniques will be widely available to encode existing legal records for many years yet.

However, we argue that significant benefit to searching and analysis of legal documents can be gained even if only a minimal encoding of legal semantics is applied to cases. This paper presents such a minimal encoding, and discusses how to map more complex deontic concepts into it.

Inference of deontic state at points in time is performed using an implementation of the Event Calculus [29]. This implementation operates over data stored using RDF [33]. More specifically, we use RDF named graphs [34, 8].

This paper is organised as follows. Section 2 describes the semantic web technology we employ. We then provide a brief introduction to the simplified event calculus in section 3. An overview of related research is presented in section 4. Section 5 introduces our minimal deontic ontology. A worked example is described in section 6. Finally, section 7 provides concluding remarks.

2. RDF NAMED GRAPHS

The Resource Description Framework (RDF – [33]) provides a straightforward, expressive means for knowledge representation. Each RDF statement has a subject, a predicate and an object. The instances of concepts need names that are defined with global scope and decentralised ownership. The World Wide Web Universal Resource Identifier (URI – [6]) system is used to meet both requirements. URIs contain (among other parts) a domain name portion, e.g. `vt.edu`, and a local path. Domain names are owned by particular organisations and can be used to locate the servers of an organisation unambiguously within the Internet. The local paths are under the control of the owner of the domain name, and thus can be defined at will. URIs themselves are just names, but by convention their global lookup process can be used to confirm they are genuine – for example returning human or machine readable information should the URIs be requested using a web browser.

Each subject and predicate of an RDF statement is a URI,

and statement objects can be either a URI or a literal value. The example below indicates that the title of RFC 3870 is "application/rdf+xml Media Type Registration". We use 'S', 'P' and 'O' to label the RDF statement's subject, predicate and object respectively. Note that the concept of "title" in this RDF statement is drawn from the Dublin Core ontology.

```
S: http://www.ietf.org/rfc/rfc3870.txt
P: http://purl.org/dc/elements/1.1/title
O: "application/rdf+xml Media Type Registration"
```

This paper does not focus on implementation-level RDF concerns, indeed our prototype evaluations are written in Prolog and use an RDF-compatible, functional term representation of our data. Our requirement was for an encoding of directed graphs with named edges: RDF can easily encode such structures.

Unlike the ideal global semantic web, however, we are encoding data that is highly subjective, transactional and possibly contradictory. Clearly court cases frequently arise from parties having conflicting beliefs. Also, the submissions received within legal proceedings are sets of statements presented together, and thus are transactional. To support representation of this sort of data, we use RDF named graphs [34].

RDF named graphs introduce an extra term to each statement, creating a quad instead of a triple. Three of the elements of this quad are the usual elements of RDF triples. The fourth element allows grouping of sets of statements. For example, we could rephrase the above RDF triple in named graph form ('G' labelling the graph).

```
G: http://vt.edu/people/abra/claim/20070501a
S: http://www.ietf.org/rfc/rfc3870.txt
P: http://purl.org/dc/elements/1.1/title
O: "application/rdf+xml Media Type Registration"
```

By referring to this RDF statement using the graph name, we can reason about an individual at Virginia Tech's claim that RFC 3870's title is as shown. It is worth pointing out that there is some interplay between RDF named graphs and the implementation of the RDF store in use, however. Strictly, if the RDF Schema [35] semantics are added to storage and inferencing components, the notion of closed lists can be used to describe named graphs. However the URIs that describe the elements of any such named graph can only build the graph's statements through reification, so the representation ends up somewhat bulky and inconvenient. We note that this is a similar but more extreme version of the situation with typed RDF literals, which can also be implemented using standard RDF statements and untyped literals provided that the RDF engine understands the conventions employed.

There is no restriction on RDF statements using graph-name URIs in their subjects, objects and predicates (although the former two contexts would probably be more intuitive than the latter). Indeed we employ statements about the containment of named graphs within other named graphs to effect collections of RDF statements and named graphs in a manner similar to the discrete "files" used in most computer software.

```
holds_at(U,T)       :- 0=<T, initially(U),
                       \+ clipped(U,0,T).
holds_at(U,T)       :- happens(E,Ts), Ts < T,
                       initiates(E,U,Ts),
                       \+ clipped(U,Ts,T).
holds_for(U,T1,T2) :- holds_at(U,T1),
                       \+ clipped(U,T1,T2).
clipped(U,T1,T2)    :- happens(E,T), T1<T,T<T2,
                       terminates(E,U,T).
initiates(E,U,_)    :- initiates(E,U).
terminates(E,U,_)   :- terminates(E,U).
terminates(E,F=_,T):- initiates(E,F=_,T).
```

Figure 1: A basic Prolog implementation of the Simplified Event Calculus

In this paper we use the following shorthand for statements that belong to named graphs:

```
graphname: { subject | predicate | object }
```

Each of graphname, subject, predicate and object represents a URI unless it is quoted. Note that we will sometimes include attributes in parentheses on these URIs. Again this is shorthand for URIs that have particular relationships (e.g. sequencing) recorded in RDF statements we do not provide.

3. THE SIMPLIFIED EVENT CALCULUS: A BRIEF INTRODUCTION

The Simplified Event Calculus can be described succinctly in Prolog (for other formulations see [31, 26]), as shown in figure 1.

In addition to some basic inference rules, the core concepts in the Simplified Event Calculus are the application-specific events, fluents and rules. Events are named instantaneous happenings that occur at a particular point in time. Fluents are named half-open intervals in time that describe a state of affairs holding over that time period.

Application-specific rules provide the connection between events and fluents. In the simplest case this will be done with instances of `initiates/2` and `terminates/2` facts that indicate respectively what fluent will be initiated or terminated by the occurrence of some event.

Both events and fluents may be parametrised. This is particularly straightforward in Prolog since atomic terms used to represent the names of either can simply be replaced by functional terms.

Often time will be treated as discrete (i.e. integer), as this avoids any problems with equality considering computers' lack of floating-point precision. The semantics of the time metric will probably be useful for any given application environment, but from the perspective of the Event Calculus inference mechanisms, all that is required is a partial order of subsets of events.

4. RELATED WORK

Amongst the first to attempt to distill fundamental legal conceptions, was jurisprudential theorist Wesley Newcomb Hohfeld [19], who identified notions such as duties (obligations), privileges, powers, and immunities. However, Hohfeld did not attempt to formalise these notions, and initial formalisms can be traced to deontic logics – logics of obligation and permission – from von Wright onwards [32, 23]. Subsequent authors have also formalised calculi for violation [5, 9] and power [20]. For a full review of various deontic and related logics, see [1].

Much prior work has been done on adding dynamic features to deontic logics: this includes work on temporal, dynamic, defeasible, and event-based variants, as well as markups and languages for contract representation and monitoring.

Various authors have investigated temporal deontic logics. Dignum et al. [10, 11, 12] observe that Meyer's deontic dynamic logic [23] is capable of expressing only immediate obligations where the action should be performed as the next action. They argue for an extension that caters for obligations to perform an action as soon as possible, before a deadline (relative or absolute time condition), or periodically. Examples of obligations specifying a deadline include obligations to 'pay within 30 days of purchasing', 'pass within 1 year of enrolling', 'order before stock is too low', or 'repair the roof before the October rains start'. Periodic obligations may include 'pay employees between 25th and 30th of every month' or 'order whenever available stock is between 6 and 10 units'. Dignum et al. model the dynamic system in terms of a purely theoretical formal mathematical Kripke structure, providing no practical implementation. Initially [12], these structures constrained their framework by forcing the impractical assumption that all actions take the same amount of time. Later [10], they propose that an assumption of instantaneous begin- and end-actions may be appropriate, but requires further investigation.

Bons et al. [7] introduce an 'epistemic dynamic deontic temporal logic', using the notation $[\beta]O\alpha$ to indicate that state-of-affairs β causes a transition to the state-of-affairs $O\alpha$ where α is obliged. As an example—

$$[promise(r_1, r_2, \alpha)]O_{r_1 r_2}(\alpha)$$

—means that 'after r_1 promises to perform α to r_2, then r_1 is obliged towards r_2 to actually perform the action α'. State history is not kept and contradictory rules and conflict resolution are not attended to. For instance, subsequent imposition of a law to the effect that 'only promises made by legally competent parties lead to obligations' could not be introduced.

Eiter et al. [13] introduce a *do* operator for the absence of a mechanism for triggering state changes in standard deontics. Bons et al. [7] proceed similarly; their $DO(\alpha)$ operator entails that α will hold at the next moment; and $DONE(\alpha)$ entails that α held at the very last moment (α was the very last action performed). For instance we can indicate indicates that the very last action was John paying \$100 using $DONE(\text{pay}(\text{John}, \$100))$. This operator is troublesome as it needs to be reset after each system event to reflect which action was last performed, and problematic race conditions can be expected in the determination of exactly which was the last action performed. No action history is maintained and logical errors arise as DONE(α) is overwritten.

A number of authors have built defeasible reasoning extensions of deontic logics – for example [28], and others – see [18] for a review.

The event calculus has also been previously applied to formalising the dynamic aspects of deontic relations (e.g. [22, 15]).

Annotation of business contracts using a markup language has also been investigated – for example, [17, 16], extended RuleML for this purpose. Other languages intended for contract monitoring e.g. [24] have also been suggested.

Our approach differs from the approaches discussed above in a number of ways:

Firstly, in previous approaches, obligation, prohibition, and permission are often regarded as unitary concepts, and their nuances are frequently overlooked. We delve deeper into different types of obligation, violable and inviolable prohibitions, and violable and inviolable permissions (see Section 5 of this paper).

Secondly, conventional approaches typically do not offer the automatic instantiation of individual identified obligations, from general obligation clauses. For example, in [17], the representation can capture that clause 6.1b specifies that "the purchaser is obliged to pay the set price for the good". However, active features are not provided that, for instance, automatically creates obligation 124, of John ($ID = 23$) to pay James ($ID = 26$) a sum of \$27.

Thirdly, because traditional approaches omit features that automatically create obligation instances, there is limited expressivity for annulments of individual obligations [2], and instead only general expressions of norm defeasibility can be provided. For instance [17], allows expressions of the form "clause 6.1c overrides clause 6.1b" (where 6.1c is a written agreement of Standard Terms & Conditions, and 6.1b is a Purchase Order). Such general override expressions are useful, but lack the ability to express specific conditions (including evidence and law) that result in the annulment of specific, identified obligation instances, in particular, well-described circumstances: for instance, clause 6.1c of the Standard Terms & Conditions may not override clause 6.1b (the Purchase Order) if the Standard Terms & Conditions were not prominently displayed and pointed out to that particular purchaser – in that instance, obligation 124, which is an individual obligation that arose from the general obligation in 6.1b, would be binding.

Finally, conventional approaches assume singular interpretations of the deontic state and do not permit different (subjective) interpretations of the status of the obligation, by different stakeholders, at different times, or using different justifications (e.g. evidence and systems of law). For instance, in the light of certain evidence, prior case law, and relevant legislation presented by the attorneys, High Court Judge Stevens may view John's obligation (obligation $ID = 124$)

as binding, whereas Supreme Court Judge Thomas, who considers additional pleadings (evidence, prior case law, and legislation) may view John's obligation as annulled. Our approach specifically caters for a plurality of interpretations, and allows the selection of a specific interpretation as the interpretation accepted by a given party at a given point in time.

In this paper, as in our previous work [2], we are not aiming to build complete and conflict-free logics, since these are not capable of capturing the rich, conflict-ridden real world scenarios and legal arguments. Rather, we are defining structured mechanisms for storing subjective, and possibly contradictory, viewpoints. Huhns and Singh [25] remark that a system that supports Hohfeld's conceptions could represent contracts and would be useful for defining and testing the compliance of agent interactions to norms. We do not claim to achieve the ambitions goals of testing agent compliance. Instead, we have defined an extended, and structured representation of Hohfeld's conceptions, and a mapping between them, which we will show facilitates richer searching of legal arguments that have been annotated using our mechanisms.

There have been numerous commercial system created for the management of contracts. See [27] for a review and criticism of these. There have also been some commercial attempts at annotating legal texts, e.g. RuleBurst from RuleBurst Corporation [30], previously known as STATUTE Expert / SoftLaw. The primary purpose of RuleBurst has been the administration of government statutes in order to correctly infer entitlements of citizens to prescribed social security, veterans, and housing benefits, and for assisting with regulatory compliance checking, for instance, in the financial sector. RuleBurst aims to provide software solutions for prevention, detection, and cure, of breaches in regulation. Our goal is simply to provide richer annotation primitives and compounds, for the purposes of improving the fullness of information held in annotations.

5. A MINIMAL DEONTIC ONTOLOGY

This section explores how a number of deontic concepts can be mapped into a simplified ontology. We emphasise that our simplification loses information in this mapping process. Nonetheless, we hope to demonstrate that it still facilitates broad inferences to be made automatically regarding the interrelationships between clauses in legal documents.

We first describe the deontic concepts that have principals as their subject and actions or states of affairs as their object. Note that the management of embedded propositional content in the target of these terms relates to work done by Kimbrough on disquotation [4, 3, 21].

Obligation. Broadly, an obligation describes a perceived requirement for a principal to react in some way in appropriate contexts. In more RDF-oriented terms, an obligation is a statement of justification that can be linked to actions taken by a principal.

The target of obligations is essentially a statement of ideality, so it is important to distinguish its terms from fact. As described above, RDF named graphs are used to do so.

Affirmation. Affirmations are declarations of belief by entities that the current world state has certain properties. As we explore in other publications [14], we aim to avoid assumptions of objective truth in our analyses.

Each of the above deontic statements have a variant with a negated object. We need to include such expressions in order to make the representation sufficiently complete. In general, negation can significantly complicate inferencing processes, so we instead add explicit negated versions.

Prohibition. A prohibition can be approximated by an obligation to not do something.

Annulment. The affirmations of negative results are *annulments*. Annulments are key to inferring outstanding obligations. Revision of the requirements otherwise implied by obligations can be made through annulments – they are a mechanism to cancel out obligations, prohibitions, violations, affirmations and other annulments. The process of determining annulment is recursive. That is to say that an annulled obligation for which the annulment is annulled needs to be viewed as an outstanding obligation.

The above terms all relate to ideal-world situations. We also need a notion of violation.

Violated-by. Instances of violated-by predicates are used to describe situations in which an outstanding obligation is perceived not to have been satisfied. As for the other deontic statements, the descriptions of these non-ideal situations can operate at different generality levels. Specific violations will reference an obligation instance.

Each instance of one of these core deontic concepts is represented by a single RDF statement. The need to collect graphs of information regarding the object of such statements means we expect the statements to belong to, or refer to named graphs.

The RDF statements representing instances of the above statement types are declared within our software as events (§3). As required by the event calculus, each such event will be recorded as occurring at a point in time. Note that these times indicate when statements are recorded, and do not relate to the times of any events those statements might refer to. The named graph containing the event can, of course, be annotated with further attributes of context (e.g. location).

We also need to incorporate some event calculus concepts into our model. The actions that affect deontic state are recorded as events. This includes events triggered by the expiry of deadlines. The world state is measured by querying fluents.

Our model can be summarised as:

1. Affirmations define world states.
2. Obligations and prohibitions express constraints on ideal world states,
3. Events can cause non-ideal world states expressed as violations.
4. Annulments can return non-ideal world states to ideal world states.
5. Events can activate annulments.

Expressed in this way, it is hopefully clear that notions such as "satisfaction" can be represented using annulment. While humans would of course make a semantic distinction between satisfaction and annulment, the effect on outstanding obligations (from a machine perspective) is very similar.

Each obligation, violation, affirmation and annulment will be used as the predicate in some number of RDF statements: they all have a subject and an object. Importantly, the description of these entities can be at very different levels of generality. For example, the encoding of a prescriptive workflow might contain both statements over "types" of subjects and objects, as well as specific statements regarding well defined entities. These specific statements usually can be formed by inference given the general statements and enough information about the world state. To allow statements to be made about entities that have not yet been defined, there will be the need either to agree on the terms used to describe concepts in advance, or to apply RDF inference tools to discover appropriate isomorphisms.

We now proceed to examine a number of deontic concepts and indicate how they can be reduced into the core set of terms described above. The information mapping loses information (e.g. the subtlety of different words), but is only intended to facilitate computerised assistance of humans searching and examining the actual source text. This list is intended to be indicative of how the mapping is implied, but it is not completely comprehensive – some concepts will need approximation to fit into the mapping.

5.1 Obligation

Since obligation is one of the core concepts we use, here we examine some of the different forms of obligation used in legal material and describe how the are mapped in particular.

5.1.1 Conditional obligation

The base form of obligation we use correlates to an unconditional obligation. *Conditional obligations* can be represented by coupling a base, unconditional obligation instance, with an appropriate annulment instance.

5.1.2 Different obligation generality levels

General, essentially parameterised obligation "factories" can be used to create instances of specific obligations. The latter are sometimes referred to as *identified obligations* [2]. In contrast, *prima facie obligations* are likely to be represented with highly generalised subjects and objects. RDF statements will need to link the particular entities within a scenario with the subject and object described. Similarly *personal obligations* require that an identified subject performs the necessary actions for their satisfaction.

Limited forms of inference can be performed over these different forms of obligation without having to analyse these distinctions, through being able to separate such annotations into separate RDF statements.

5.1.3 Interrelated obligations

There are numerous forms of obligation that imply sequencing. For example a sequence of steps in a workflow (series or parallel), or so called *joint obligations*, in which any member of a group can perform the necessary actions to avoid violations occurring. Note that in this case all group members will be seen to be in violation if the obligation is violated.

5.2 Prohibition

Prohibition is of course a commonly used deontic principle. As discussed above, prohibition is a core statement type: essentially an obligation not to do something. This most directly relates to the notion of a *Violable prohibition* in deontic literature. In this case, a violation of the prohibition occurs when, for example, a particular prohibited action is performed.

An *inviolable prohibition* (also known as immunity or disability) on the other hand, is a statement about the limitation of power of a subject. For example, say some hypothetical retail store policy declares something to the effect that "sales staff cannot extend the warranty period on product X". If a sales staff member subsequently asserts an extended warranty period to a customer, the statement is annulled. That is, the sales staff member never had the power to make the assertion in the first place. Note that in some jurisdictions, if the sales staff member makes the warranty extension assertion to a customer without showing them the company conditions then common law says the warranty is actually extended.

We encode inviolable prohibitions in our simplified form by declaring annulments of the potential affirmations of the relevant type.

5.3 Positive and negative statements

Our mapping looses the connotational distinction between *assertions* and affirmations, but hopefully it is clear they have similar semantics in terms of their impact on the world state. Affirmations are a core item of our ontology. Note that this word was chosen to avoid the computer programming connotations associated with the term "assertion".

As discussed above, *annulment* is also an important deontic concept, and is within the core ontology. Its meaning is an affirmation of "not X".

The term *satisfaction* is not within the core ontology however. As previously discussed, we emphasise that satisfaction causes a cancellation of (potential) outstanding obligations, and thus can be represented using annulment.

Satisfaction adds one extra condition compared to straight annulment: it is annulment of a clause that is not violated.

Intuitively the status of an obligation can either be violated but not annulled, violated and annulled, satisfied, or outstanding.

5.4 Violation of obligations

Violation-by is another core ontology member. The "-by" suffix is used to invert the subject and object sense for RDF statements of this type. A policy object is the subject of such statements, and the conditions of the violation are defined by the object.

Note we ensure that violation is recorded as an event in itself. Noticing violations makes sense in a human context. It normally is not associated with computer inference, but we believe that we need it to be, given that our infrastructure can look at situations from numerous independent perspectives.

Fully annotated instances will record both the origin of the violation in terms of the events (evidence) that caused it, as well as the documents / utterances (provenance) that defined the violation as arising under those circumstances.

5.5 Powers and permission

The notion of *power* has two senses. In one sense power describes the relative weight of different principals' affirmations. So if party A affirms "X" and party B affirms "not X", the party with the power to make such statements will be sought out to resolve the conflict.

The other sense of powers is as a form of automatic affirmation. So some action can trigger an affirmation because of a given power.

In either case, the partial order of overrides will need to be established to resolve any conflicts. So, coupled with appropriate override orders, a power can be seen as a conditional affirmation, along with a prohibition from others contradicting the affirmation when viewed as misleading from the perspective of that institution. For example, Catholic priests have the power to proclaim two people to be married, but an unordained person does not. A proclamation by an unordained person would be regarded as both a violation and as being ineffectual (i.e. annulled), by the Catholic institution (although of course other institutions may take a different view).

Finally *permissions*, like prohibitions, can be divided up into two classes: *violable permissions* and *inviolable permissions*.

An inviolable permission (also known as a *privilege*) can be represented as annulment of violations arising from prohibitions. For example, if A is permitted to do X, but B prohibits this action, then violations against B's prohibition will not be accepted.

A *violable permission* is an obligation on other parties not to interfere with certain actions of the other party. Such forms of interference will lead to violations.

6. CASE STUDY

To explore the practical use of our framework, we encode a subset of the clauses contained in a legal case: the ruling by the High Court of South Africa in case number AC166/2003, delivered on 15th Dec 2005.

The case can be retrieved by the interested reader from `http://law.sun.ac.za/` for the sake of extra context and/or to verify the results and analysis in this paper.

6.1 Summary of the case

The conflict is between plaintiff Freshgold SA Exports (Pty) Ltd ("Freshgold") and defendant Maritime Carrier Shipping GmbH and CO ("Maritime").

Freshgold's business includes the export of fruit. In this case, a shipment of fruit was to be transported by Maritime from Cape Town to Rotterdam in a 40 foot refrigerated container. It was agreed by both parties verbally that the fruit needed to remain below -0.5 degrees centigrade during its transport.

Unfortunately, after the container was loaded onto its intended transport vessel (Marine Vessel "Grey Fox"), it malfunctioned. It was offloaded from that vessel for repairs. It was subsequently reloaded onto Marine Vessel "Amber Lagoon", and reached Rotterdam. However, the fruit had suffered from decay and mould and had to be sold at reduced prices.

Freshgold sought compensation for the damaged goods from Maritime claiming that the damage occurred due to Maritime violating the agreement between them that the fruit be kept at an appropriate temperature.

It was agreed by both parties, however, that the damage to the fruit most probably occurred in the time when it was between the two vessels. Unfortunately for Freshgold, Maritime attached its standard terms and conditions to the bill of lading issued for the transport. In the terms and conditions, there are two clauses of particular interest:

> Clause 5: The carrier shall be under no liability whatsoever for loss of or damage to the goods, whosoever occurring if such loss or damage arises prior to loading onto or subsequent to discharge from the vessel.
>
> Clause 16. The carrier may at any time and without notice, transfer the goods from one vessel to another

Freshgold's representatives claimed they could not believe it would be the case that Maritime would not be responsible for damage caused by a defective container. While the judge acknowledged that this was indeed a high point in the plaintiff's case, the judge also emphasised that Freshgold would have expected to receive a bill of lading and that the bill of lading might well include the standard terms and conditions of the shipper. Thus, the plaintiff was responsible for not having negotiated an arrangement that would preclude clause 5 protecting Maritime from responsibility during the need to exercise clause 16. *Caveat emptor...*

Discourse	Description
0	Written record of the judge's decision
1	Record of the oral agreement between the plaintiff and the defendant
2	Common cause evidence
3	Standard terms and conditions of the defendant
4	Bill of Lading for the shipment of the 40 foot refrigerated container, containing fruit, from Cape Town to Rotterdam
5	Plea by the plaintiff
6	Plea by defendant

Table 1: Discourses in the example case

```
case(0): { p | affirms | discourse(5)
case(0): { discourse(5) | type | discourse }
case(0): { discourse(5) | contains | d(5,1) }
case(0): { discourse(5) | contains | d(5,2) }
case(0): { discourse(5) | contains | d(5,3) }

d(5,1): { d(1,2) | annuls | d(3,5) }
d(5,1): { d(5,1) | context | discourse(1) }

d(5,2): { d(1,1) | violated-by | d }
d(5,2): { d(1,2) | violated-by | d }

d(5,3): { d | obliged | damages }
d(5,3): { d(5,2) | evidence-for | d(5,3) }
```

Figure 2: Clauses encoding the plaintiff's plea

6.2 Encoding of the case

The encoding of the case is done in a number of stages. We use the term *discourse* to describe the named graphs that collect the RDF statements that make up the salient parts of evidence submitted by the plaintiff and the defendant, as well as the decision delivered by the judge. As mentioned in section 2, we employ an orthogonal ontology of graph relationships, in particular the "contains" predicate.

```
sw(0): { case(0) | case-title | "High Court of South
         Africa Case Number AC166/2003" }
sw(0): { case(0) | case-delivered | "2005-12-15" }
sw(0): { case(0) | contains | discourse(0) }
sw(0): { case(0) | contains | discourse(1) }
...
```

The discourses extracted from this example case are enumerated in table 1.

6.3 The plea from the plaintiff

We begin by examining a subset of the clauses encoding the plaintiff's plea, shown in figure 2. All of the terms in the listings are placeholders for URIs. The form `d(`x`,`y`)` indicates the yth URI of discourse x. The terms `p` and `d` refer to the plaintiff and defendant within this case.

In figure 2, the named graph `d(5,1)` declares that that plaintiff sees clause 5 (that the carrier is not liable for damage when the goods are not on a vessel) of the standard terms and conditions as being annulled. The reason is a clause

```
case(0): { j | affirms | discourse(0) }
case(0): { discourse(0) | type | discourse }
case(0): { discourse(0) | contains-graph | j(1) }
...
case(0): { discourse(0) | contains-graph | j(9) }

j(1): { ap | agent-of | p }
j(1): { ad | agent-of | d }
j(1): { ap | affirms | discourse(1) }
j(1): { ad | affirms | discourse(1) }

j(2): { discourse(1) | type | oral-agreement }

j(3): { discourse(2) | type | common-cause }
j(3): { p | affirms | discourse(2) }
j(3): { d | affirms | discourse(2) }

j(4): { discourse(4) | provided-with | discourse(3) }

j(5): { d(5,1) | annuls | d(3,5) }
j(5): { d(5,1) | annuls | d(3,16) }

j(6): { r1 | annuls | j(6b) }
j(6b): { discourse(4) | type | contract }

case(0): { r1 | type | document-reference }
case(0): { r1 | section | discourse(0) }
case(0): { r1 | page | 7 }

j(7): { p | obliged | discourse(3) }
j(7): { j(4) | evidence-for | j(7) }

j(8): { p | obliged | d(3,5) }
j(8): { p | obliged | d(3,16) }

j(9): { j | annuls | d(5,1) }
j(9): { j | annuls | d(5,3) }
j(9): { j(8) | evidence-for | j(9) }
```

Figure 3: Clauses encoding the judge's decision

of the verbal agreement between the parties (`d(1,2)`). As noted above, the functional form of the above URI terms is only for reader convenience.

In addition to the core deontic predicates, this named graph also indicates that `discourse(1)` is the `context` of `p`'s annulment, i.e. the annulment is only related to this particular set of circumstances, and not the defendant's terms and conditions as applied to other agreements.

The second named graph, `d(5,2)` indicates the plaintiff's affirmation that two clauses of the verbal agreement between them have been violated by the defendant.

Finally, graph `d(5,3)` indicates that the plaintiff believes that the defendant is obliged to pay damages (although we ignore the statements to which `damages` refers).

6.4 Conclusion from judge

We now turn our attention to encoding the result of this court case, as presented by the judge. A subset of the necessary clauses are shown in figure 3.

The first few statements indicate that the judge (`j`) affirms the content of the named graph `discourse(0)`. We use this

pattern to simplify a party making a collection of affirmations. Statements are also included that specify the named subgraphs included with this particular discourse.

The named graph `j(1)` sets up the parties `ap` and `ad` as agents of the plaintiff and defendant, and indicates that the items in discourse 1 are agreed between them both. For completeness we include `j(2)`, it indicates that the judge noted that the initial agreement between the two parties was oral. A more comprehensive annotation would need to include details such as the time and location of this statement, although we only need a partial ordering of events in this paper to perform our analyses.

The common cause evidence is discussed in named graph `j(3)`. It indicates that both the plaintiff and the defendant affirm the statements it contains.

The first significant item of the case is encoded in `j(4)`. The judge noted that the bill of lading (discourse 4) included the defendant's standard terms and conditions (discourse 3).

As previously discussed, the plaintiff chose to annul certain clauses of the defendant's terms and conditions. This is indicated in `j(5)`. The judge notes that the bill of lading itself is not a contract in `j(6)`, although as this is not directly connected to the outcome of the case, we leave it as an external reference to source document material.

After weighing up the evidence, the judge concludes that the plaintiff is bound by the terms and conditions, since they should have expected that clauses similar to 5 and 16 would be included. This is encoded in `j(7)`. This leads quickly to the judge emphasising that the plaintiff is obliged by the two relevant clauses in the standard terms and conditions (`j(8)`), and that the judge annuls both the plaintiff's dismissal of the clauses of the terms and conditions, and also the plaintiff's belief that the defendant owes them damages (`j(9)`).

6.5 Analysis of case

In this section we demonstrate a "quick and dirty" implementation of the event calculus providing predicted answers over the case. Of course real deployments would require a more comprehensive implementation – in particular the use of RDF querying technologies should be better combined with the event calculus engine.

We used a simple shell script to extract the (named graph) RDF statements we have discussed from the LaTeX source of this paper. These lines were transformed into Prolog statements (`data.pl`) that are loaded into the SWI-Prolog source file presented in figure 4.

Note that figure 4 introduces the predicate `d_holds_at`. The notion here is that a fluent *deontically holds* if it holds and it is not annulled by a fluent that deontically holds itself. In this paper we create a partial order of events for the event calculus engine from the order in which the named graphs are declared by the parties in the court case. The time units chosen achieve the required partial ordering, but are otherwise arbitrary.

At time unit 15, the plaintiff's plea is fully presented. From time 42 onwards, the judge's decision has been fully presented. Predictably we get the following Prolog dialogue:

```
:-ensure_loaded(data).
:-dynamic holds_at/2.
:-ensure_loaded(event_calculus).

d_holds_at(U,T) :-
        holds_at(U,T),
        \+ d_annulled_at(U,T).

d_annulled_at(U1,T) :-
        annulled_by(U1,U2),
        d_holds_at(U2,T).

annulled_by( annulment(G1,S1,O1), annulment(_,_,G1) ).
annulled_by( obligation(G1,S1,O1), annulment(_,_,G1) ).

initiates(rdfng(G,S,annuls,O), annulment(G,S,O) ).
initiates(rdfng(G,S,obliged,O), obligation(G,S,O) ).
```

Figure 4: Prolog code to test inference

```
?- d_holds_at( annulment(d(5,1),d(1,2),d(3,5)), 15).
Yes
?- d_holds_at( annulment(d(5,1),d(1,2),d(3,5)), 42).
No
?- holds_at( obligation(d(5,3),d,damages), 50).
Yes
?- d_holds_at( obligation(d(5,3),d,damages), 50).
No
```

In other words, at time 15 named graph `d(5,1)` indicates that a clause of the spoken agreement (`d(1,2)`) annuls clause 5 of the standard terms and conditions (`d(3,5)`). This is the plaintiff's perspective. By time 42, this annulment has been annulled itself. Similarly, at time 50 the plaintiff's named graph `d(5,3)` indicates the belief that the defendant (`d`) should pay damages (externally referred to via the URI `damages`), although this obligation does not deontically hold at that time, as it has been cancelled out by the judge's affirmations.

Thus the infrastructure we have described can facilitate basic inference of the world state at different times during the progress of a case. It can also indicate the independent believed world states as viewed from different participants. It is hoped it would be straightforward to appropriately annotate cases, and that the extra inference mechanisms could increase the expressiveness of queries over legal databases.

Note that we do not claim to be able to infer whether the judge reached a "correct" conclusion in the case. Our only goal is to better annotate the transcript. Whether the judge's conclusion is correct would be the subject of complex legal arguments, citing diverse, and sometimes previously uncited, statutes, precedents, and evidence. Also, irrespective of its procedural correctness, the judge's findings are still upheld, until a higher court decides otherwise, and it is of little avail for us to dispute his findings, since only he has the power to proclaim them, and only higher institutions have the power to annul them.

7. CONCLUSION

This paper introduces our initial work on a simple deontic ontology for annotation of legal documents. We use RDF named graphs as an expressive and flexible mechanism for subjective knowledge representation. An event calculus implementation is discussed that can perform queries over annotated legal data. We then indicate how these technologies might be employed in the approximate encoding of a South African High Court case.

The requirement to perform logic programming reduces the real world applicability of our current work, although we are hopeful that aspects of the annotation process can be automated.

Although our representation loses information in the mapping process, we believe that simplification of semantics will increase the scalability of document annotation processes. We are hopeful that when used alongside electronic versions of the source material, the technologies we have presented might help initial search and analysis of collections of legal documents.

8. REFERENCES

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

[2] Alan S. Abrahams and Jean M. Bacon. The life and times of identified, situated, and conflicting norms. In *Procedings of the Sixth International Workshop on Deontic Logic in Computer Science (DEON'02), Imperial College, London, UK*, May 2002.

[3] Alan S. Abrahams and Jean 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.

[4] Alan S. Abrahams and Steven O. Kimbrough. Treating disjunctive obligation and conjunctive action in event semantics with disquotation. *Wharton Business School Working Paper Series*, 2002.

[5] Alan R. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67:100–103, 1958.

[6] T. Berners-Lee. RFC 2396: Uniform Resource Identifiers (URI). Technical report, MIT, 1998.

[7] R.W.H. Bons, F. Dignum, R.M. Lee, and Y-H. Tan. A formal analysis of auditing principles for electronic trade procedures. *International Journal of Electronic Commerce*, 5(1), 2000.

[8] Jeremy J. Carroll. Signing RDF graphs. In *Proceedings of the International Semantic Web Conference*, Florida, USA, October 2003.

[9] P. d.Altan, J.J.Ch. Meyer, and R.J. Wieringa. An integrated framework for ought-to-be and ought-to-do constraints. *Artificial Intelligence and Law*, 4:77–111, 1998.

[10] F. Dignum and R. Kuiper. Combining dynamic deontic logic and temporal logic for the specification of deadlines. In *Proceedings of the 30th Hawaii International Conference on Systems Sciences*, Hawaii, 1997.

[11] F. Dignum and R. Kuiper. Specifying deadlines with dense time using deontic and temporal logic. *International Journal of Electronic Commerce*, 3(2):67–86, 1998–1999.

[12] F. Dignum, H. Weigand, and E. Verharen. Meeting the deadline: on the formal specification of temporal deontic constraints. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, pages 243–252, 1996.

[13] T. Eiter, V.S. Subrahmanian, and G. Pick. Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, 108(1-2):179–255, 1999.

[14] David M. Eyers and Alan S. Abrahams. Orthogonal truths, electronic beholders and the event calculus. In *Workshop on Formal Modelling for Electronic Commerce*, Palo Alto, USA, June 2007. To appear.

[15] Andrew D. H. Farrell, Marek Sergot, Mathias Salle, and Claudio Bartolini. Using the event calculus for tracking the normative state of contracts. *International Journal of Cooperative Information Systems*, 4(2–3), 2005.

[16] G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2–3), 2005.

[17] G. Governatori and A. Rotolo. Modelling contracts using RuleML. In *Proceedings of Jurix 2004: Legal Knowledge and Information Systems*, pages 141–150, Amsterdam, 2004. IOS Press.

[18] G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *Proceedings of the Tenth International Conference on Artificial Intelligence and Law*, pages 25–34, New York, NY, USA, 2005. ACM Press.

[19] Wesley N. Hohfeld. *Fundamental Legal Conceptions as Applied in Judicial Reasoning*. Greenwood Press Publishers, 1978.

[20] Andrew J.I. Jones and Marek Sergot. A formal characterisation of institutionalised power. *Journal of the Interest Group in Pure and Applied Logic*, 4(3):427–443, 1996.

[21] Steven O. Kimbrough. Reasoning about the objects of attitudes and operators: Towards a disquotation theory for the representation of propositional content. In *Eight International Conference on Artificial Intelligence and the Law (ICAIL 2001)*, St Louis, Missouri, May 2001.

[22] R. Hernandez Marin and G. Sartor. Time and norms: a formalisation in the event calculus. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, pages 90–99, New York, NY, USA, 1999. ACM Press.

[23] John J.C. Meyer and Roel J. Wieringa. *Deontic Logic in Computer Science*. John Wiley & Sons Ltd, 1993.

[24] Z. Milosevic, S. Gibson, P. F. Linington, J. Cole, and S. Kulkarni. On design and implementation of a contract monitoring facility. In *First IEEE International Workshop on Electronic Contracting (WEC'04)*, pages 62–70, 2004.

[25] Huhns M.N. and Singh M.P. Agent jurisprudence. *IEEE Internet Computing*, 2(2):90–91, 1998.

[26] Erik T. Mueller. *Commonsense Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA,

2006.

[27] S. Neal, J. Cole, P.F. Linington, Z. Milosevic, S. Gibson, and S. Kulkarni. Identifying requirements for business contract language: A monitoring perspective. In M. Steen and B.R. Bryant, editors, *Proceedings of the seventh International Enterprise Distributed Object Computng Conference*, pages 50–61, Brisbane, Australia, September 2003. IEEE Computer Society.

[28] D. Nute. Norms, priorities, and defeasible logic. In P. McNamara and H. Prakken, editors, *Procedings of the second International Workshop on Deontic Logic in Computer Science (DEON'98)*, Amsterdam, 1998.

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

[30] Ruleburst corporation. `http://www.ruleburst.com/`, 2007.

[31] Murray Shanahan. The event calculus explained. *Springer Lecture Notes in Artificial Intelligence*, 1660:409–30, 1999.

[32] G.H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

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

[34] W3C. OWL web ontology language overview. `http://www.hpl.hp.com/techreports/2004/HPL-2004-56.html`, February 2004.

[35] W3C. RDF vocabulary description language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema/`, February 2004.