Using annotated policy documents as a user interface for process management

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

***Abstract*—Natural language policy documents are frequently used as starting point for requirements capture, leading to computer systems that manage process management within organisations. Rather than modelling explicit workflow graphs of business processes, this paper proposes presentation of annotated versions of the natural language policy documents as a user interface indicating both the status of task progress and appropriate potential progress routes. A deontic adaptation of the Event Calculus is presented to monitor the normative state of policy compliance. A non-tree-based document annotation scheme is used to allow a natural language text to be linked with a logic program developed to represent its intentions. The approach is demonstrated by the encoding and presentation through a web application of a section of the United States Food and Drugs Administration regulations.**

I. Introduction

Many of the commissioned computer systems within organisations exist to guide these organisations' workers along appropriate business process routes.

Often the construction of an organisation's software will involve many interactions between requirements gathering and early software development (hence the many models of the software engineering process).

While some more subtle aspects of business process may reside only in the minds of an organisation's employees (albeit possibly such intuition providing significant competitive advantage), the majority of practices and constraints on the operations of an organisation are likely to be described using natural language. For example, contracts, internal process documentation, and the paperwork required to demonstrate compliance to regulatory bodies such as taxation offices and company registries.

Ideally, the business processes are converted into software with high fidelity. In practice, end user systems may impose significant limitations to the freedom of staff in an organisation in the interest of ensuring compliance to stated requirements. In other words, the workflow encoded is derived from a comparatively ideal-case scenario, and exceptions are dealt with in increasingly *ad hoc* ways.

Within this paper, we use the term 'policy' to mean a set of constraints over a system. A primary goal of electronic policy research is to avoid encoding policy rules implicitly and/or statically into software. Through explicit, separable policy specifications, it is more likely that the system will be able to be updated when new policy modifications are required without rebuilding the whole software system (in the ideal case policy can be updated dynamically on a running system).

This paper proposes a mechanism to keep policy at an ideal level of abstraction within job monitoring systems through a user interface grounded in the source documentation from which the requirements of the system are gathered.

The research in this paper has been applied to a section of the United States Food and Drug Administration (FDA) regulations in the domain of blood testing for infectious diseases. This document is ideal in providing a complex set of interacting obligations, prohibitions and permissions, for which the target is clearly identified (a particular blood donation).

The infrastructure presented incorporates a deontically-aware variant of the Event Calculus to monitor the state of policy compliance. This provides a significant superset of the expressiveness of workflow systems that are based on Petri Nets. In workflow graphs that use Petri Nets, incorporating support for exceptional situations may involve adding large numbers of edges, or building additional logic into the workflow engine.

The modified Event Calculus predicates, the logic-program of the source policy document, and the web server infrastructure that drives the user interface have all been implemented in SWI Prolog [1].

This paper is structured as follows. Section II presents related research work. Background information is provided in section III on the deontic concepts employed alongside a brief introduction to the Event Calculus. Section IV presents the technologies required to effect policy document user interfaces. Results produced by a prototype implementation using the FDA document as input are presented in section V. Section VI discusses some of the avenues for future research. Finally, section VII provides closing comments.

II. Related Work

The research within this paper is related to a number of different fields. Some particular aspects, such as the Event Calculus, are introduced below. There is of course a vast body of research and implementation work on managing processes within organisations. Computational approaches are often related to workflow systems (for an overview of workflow standards see [2]).

Focusing particularly on policy representations, research has been done on representing deontic state using Petri Nets [3], [4], [5] and Finite State Machines [4], [6]. Both of these technologies impose limitations on the action sequences by which policy compliance can be achieved.

This paper avoids explicit workflow encoding, instead focusing on the constraints implied by policy. This paper provides a straightforward, consistent encoding of document clauses, although one that does not aim to provide compact visualisation in the way many workflow systems do. In that sense, this work is similar to Grosof, Labrou, and Chan's representation of contracts using Situated Courteous Logic Programs [7]. However, in contrast, our logic programming representation is formalised using the Event Calculus, and represents deontic states explicitly.

In terms of user-focused frameworks that are designed to assist users to comply with policy, the authors have done work previously on two contract representation frameworks, EDEE [8], [9] and CamPACE [10]. Both of these systems take a bottom-up approach, trying to determine overall contract status from very fine grained annotation of policy clauses. The current work instead takes a top-down approach in which the semantics of document regions are encoded into logic program, but not the internal semantics of each such region.

III. Background

This section provides a brief overview of the deontic concepts we employ in this paper, and the Event Calculus that we use to track deontic state.

A. Deontic concepts

The aim of the infrastructure presented is to track the normative state of policy as actions are taken in a human-driven, but computer-assisted environment.

Natural language policy documents have a vast and rich set of terms used to indicate requirements. In the interest of logic programming representations, many of the more subtle natural language details have been elided. This is acceptable because the software only requires enough information to present the natural language clauses to the user when appropriate.

Deontic logic, the logic of obligations, prohibitions, and permissions, has a rich publication record, for example see Meyer and Wieringa [11]. Much of the literature focuses on the issues that arise when looking at logics such as Standard Deontic Logic (SDL) from a more complete (and thus future inclusive) perspective. The user interface project here would benefit from developments in deontic logic research regarding its future predictive capabilities (see section V-B). However, a basic taxonomy of deontic states has been sufficient so far to represent the document clauses we have examined.

The documents are encoded into logic programs using the limited set of states shown in table I. For example, different types of "power" are likely to be mapped into the permission predicate. Experimentation so far indicates that the computer-based representation needs to know only an approximation of the ways in which the deontic states relate to each other.

TABLE I
A minimal set of deontic states

Obligation	Some set of actions needs to be performed in the future to progress the state of affairs. The notion of *prohibition* is also covered; there is an obligation not to perform some set of actions.
Violation	Violations occur when an obligation to not do something is broken, or an obligation to do something is not done within the required time-frame.
Annulment	An annulment cancels out the effect of some other predicate (possibly another annulment). Annulments cover both the notion of exception, e.g. a document clause that overrides the effect of another clause, and the notion of satisfaction, e.g. completion of a obligatory task annuls the obligation to do that task.
Permission	Permissions are essentially a named annulment. The instantiation of a permission within a document is likely to be coupled with the explicit annulment of clauses that would otherwise be in conflict with it. For example, a permission might annul the obligations to do or not do actions specified elsewhere in that document. Moreover, the naming of permissions will allow them to be used as external reference points from other documents' clauses.

B. Event Calculus

The Event Calculus [12] is a powerful, straightforward formal modelling technique. Its most basic forms can be implemented extremely easily, and have been applied to a wide variety of software systems and research projects (see [13]). Due to lack of space the core Event Calculus predicates are not reproduced here, but are readily available in the aforementioned publications.

The basic principle of the Event Calculus is that *events* initiate and terminate *fluents*. Events are instantaneous happenings in time, and fluents are half-open time intervals that represent states of affairs. Events and fluents are both parameterised. The formulation of the Event Calculus solves the frame problem (also in [12]), and the careful specification of its core predicates ensure well-defined activity in the face of simultaneous events.

C. Blood-bank document context

The document used within the initial user interface prototype comes from the FDA Code of Federal Regulations (CFR) title 21, "Food and Drugs"; chapter I, "Food and Drug Administration"; subchapter F "Biologics"; part 610, "General Biological Products Standards"; subpart E, "Testing Requirements for Communicable Disease Agents"; section 610.40, "Test requirements". The authors' attention was drawn to this document by the work of Professor Insup Lee's group at the University of Pennsylvania [14], [15].

IV. Policy document user interface

This section introduces three technologies that help facilitate policy document driven user interfaces.

A. Event Calculus extensions

The core Event Calculus predicates allow deductive reasoning as to the states of affairs that hold at a particular point in time. In the context of this work, the core deontic states are

```
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).
```

Fig. 1. Deontic fluent inference

described in table I. However these states alone are not directly useful, because complimentary states relating to a particular clause (e.g. an obligation and its satisfaction) are maintained independently on the basis that their evidence is independent.

The results produced for document annotation apply an extra level of deontic inference to the core deontic fluents that hold at any point in time. Informally, a state is considered to deontically hold if it holds and it is not annulled by an annulment that is not itself annulled (and so on recursively). The Prolog encoding of this is shown in figure 1, in which the `T` variable represents time, and the `U` variables are fluents. Although not demonstrated here, the `annulled_by/2` predicate indicates whether one fluent would annul another (e.g. satisfactions annul obligations).

B. Document annotation

In order to display deontic feedback annotations over the source policy document, mark-up tags must be added to indicate salient document regions. The named regions used are selected for many different purposes. Most directly, clauses in the logic program representing the document are linked to named regions. There are many other framing concepts within the source documents, however, such as physical page number (in the original document), or regions for which there are recorded comments or policy updates.

Importantly, the document regions are not tree structured: they may overlap in non-hierarchical ways. Thus each aspect of document annotation must be separable. In this project, markers parameterised in two integer dimensions are used to indicate points of interest and to mark the beginning and the end of particular document regions.

It would be possible to employ XML technology for the markup required. Although tree structure cannot be used, single empty tags can be embedded as markup to indicate points, and pairs of empty tags to indicate regions within the document. The XML name-space mechanism can separate the document markup tags from a document expressed in an XML language. XML was not employed in the prototype presented, however, since in this context its benefits did not outweigh the inconvenience of its use: instead markup was discovered by searching for two instances of a known unique string surrounding the integer dimension values in a comma-separated list. This is a similar approach to MIME framing [16]: boundaries can be chosen so that the rest of the document need not be modified to escape markup.

C. JavaScript and Prolog

The prototype document viewing interface is a web application. Any interaction with a given pane can cause updates of the display on the other panes. The traditional CGI approach to this sort of application was to reload the entire page, however ideally users' browsers should maintain their viewing position within all the panes.

The more modern approach is the so-called 'Web2' one: to use JavaScript to send XMLHttpRequest requests back to the server when updates are required. When the server replies, a call-back function can dynamically modify the DOM structure of the page. Usually this is termed AJAX (Asynchronous JavaScript and XML), although given the data serialisation is Prolog, in this case AJAP seems more appropriate.

The asynchronous update request paradigm allows the control interface to respond to the user quickly, even when some of the other panes will take some time to update their display.

V. Prototype implementation

This section describes the results obtained from the prototype implementation of the document-centric user interface. In the interest of rapid development, the application is currently web-based, with the server written in multi-threaded Prolog. The interface, as shown in figure 2, consists of four panes (implemented as HTML frames in this prototype):

- **Control.** The control pane (top) allows the user to change parameters that affect all the other panes. It indicates what subject is being examined (a blood donation in this case), and the point in time t that we are examining.
- **Fluents.** The fluents pane (middle, left) lists the fluents that hold at time t.
- **Events.** The events pane (middle, right) displays the events that have occurred up until time t.
- **Document,** The document (bottom) pane is the most important user interaction element. It displays the policy document along with annotations indicating both the current state of affairs, and likely paths for future progress.

There are three types of annotations indicated within the document pane:

- **Obligations.** Based on the current point in time, and the current subject, obligations that the deontic Event Calculus framework deems to hold at time t are highlighted in blue. Each highlighted region is followed by a link arrow that requests searches for *progress paths*, which are explained in section V-B below, from this clause.
- **Violations.** Clauses that represent being in a state of violation are highlighted in red. As for obligations, violation highlights are also followed by link arrows that search for progress paths.
- **Progress paths.** If a progress path search is currently active, the potential progress clauses are highlighted in green.

The fluents in the fluent pane are also colour-coded[1]. Again, blue indicates outstanding obligations, green shows satisfaction clauses, and red indicates violations. The fluents refer to clauses via a short clause ID for visual compactness.

The clause IDs shown are digit-wise concatenations of the indexes representative of the hierarchical subclauses within the document, taking advantage of particular document features such as there being no clause list with more than ten items. However, this is only a syntactic concern: any short mnemonic can be used for the clause IDs. Moreover the IDs are hyperlinks: users clicking on them will position the document pane at the appropriate clause's location irrespective of that particular identifiers are used.

The focus of this paper is on the document annotations, rather than management of the actual event list in effect. A basic mechanism has been provided for adding and removing events via the web, but clearly a real deployment would require more accessible interface features with which to manage the event list. In terms of the current implementation, managing the events is straightforward for users that can connect to the Prolog instance running the server: each event is a Prolog fact that associate affirmations from particular users with points in time. Any means used to modify the Prolog database will effect an event list editing interface. In production environments, it is likely that the event list would be constructed dynamically from large-scale storage infrastructure such as relational databases.

One particular advantage of encoding policy documents using a symbolic logic language such as Prolog is that the valid types of known event can be scanned out of the program code directly. This could simplify the task of constructing a user interface for entering new events. For example, in the CFR document many types of blood testing events are referenced. In this case, all the possible types of test event could be automatically compiled by a user interface generator.

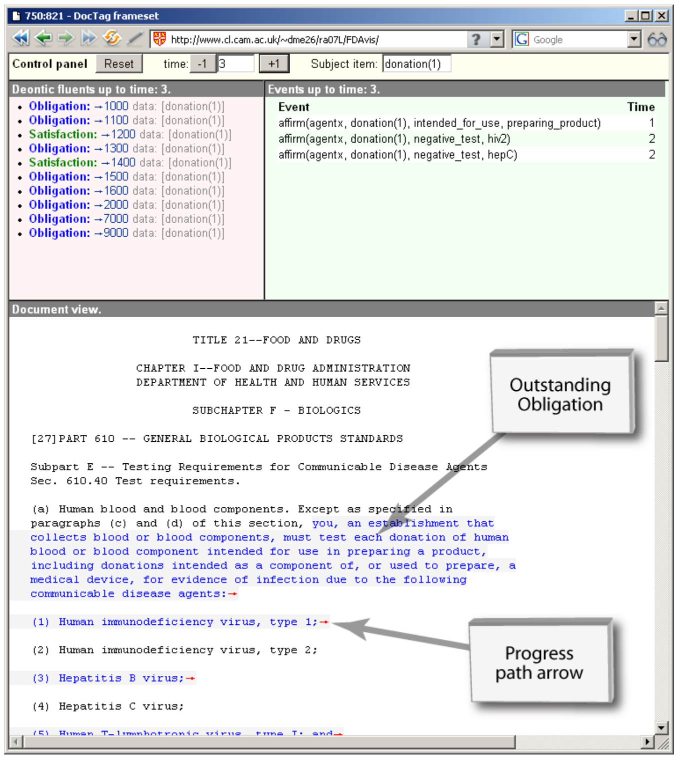
A. Example trace

In this section the actual policy context from the CFR document is discussed. Figure 2 shows an early stage (time '3') of a user 'agentx' operating on a particular blood donation 'donation(1)'.

The document pane (figure 2 lower part) is displaying the top of the CFR document (as indicated by the scroll-bar), and a number of blue highlights (obligation regions) are visible. The topmost blue obligation region was given identifier '1000', and its sub-points identifiers '1100' through '1600'. This four digit identifier system corresponds to the four levels of nesting within the original CFR document; the interested reader can confirm this correlation by acquiring a copy of the FDA document through the public portal at http://www.accessdata.fda.gov/.

The event display indicates that so far a declaration has been made that 'donation(1)' will be used for preparing a product.

[1]In the event this document is reproduced without colour, the blue document regions have a light grey background, and the green and red document regions a darker grey background



- **Obligation:** →1000 data: [donation(1)]
- **Obligation:** →1100 data: [donation(1)]
- **Satisfaction:** →1200 data: [donation(1)]
- **Obligation:** →1300 data: [donation(1)]
- **Satisfaction:** →1400 data: [donation(1)]
- **Obligation:** →1500 data: [donation(1)]
- **Obligation:** →1600 data: [donation(1)]
- **Obligation:** →2000 data: [donation(1)]
- **Obligation:** →7000 data: [donation(1)]
- **Obligation:** →9000 data: [donation(1)]

Event	Time
affirm(agentx, donation(1), intended_for_use, preparing_product)	1
affirm(agentx, donation(1), negative_test, hiv2)	2
affirm(agentx, donation(1), negative_test, hepC)	2

```
                    TITLE 21--FOOD AND DRUGS

              CHAPTER I--FOOD AND DRUG ADMINISTRATION
              DEPARTMENT OF HEALTH AND HUMAN SERVICES

                    SUBCHAPTER F - BIOLOGICS

[27]PART 610 -- GENERAL BIOLOGICAL PRODUCTS STANDARDS

Subpart E -- Testing Requirements for Communicable Disease Agents
Sec. 610.40 Test requirements.

(a) Human blood and blood components. Except as specified in
paragraphs (c) and (d) of this section, you, an establishment that
collects blood or blood components, must test each donation of human
blood or blood component intended for use in preparing a product,
including donations intended as a component of, or used to prepare, a
medical device, for evidence of infection due to the following
communicable disease agents:→

(1) Human immunodeficiency virus, type 1;→

(2) Human immunodeficiency virus, type 2;

(3) Hepatitis B virus;→

(4) Hepatitis C virus;
```

Fig. 2. Annotated screen-capture of document annotations

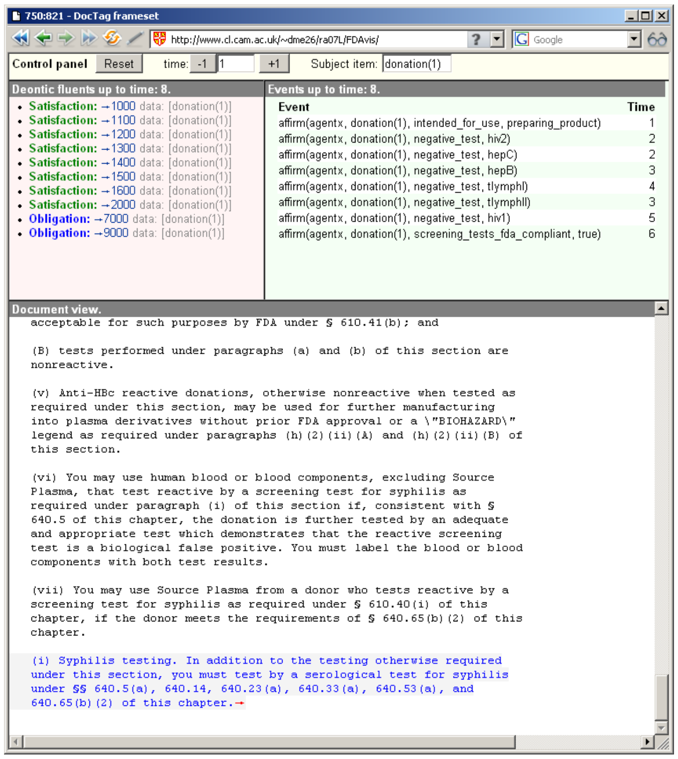
In addition two negative test results have been recorded. Indeed the lack of highlighting of points (2) and (4) near the bottom of the document view is directly because of these two test results. Note also the red progress path search arrows (discussed below) at the end of each obligation region.

The fluent pane indicates a number of outstanding obligations. Broadly speaking, obligation 1000 is the overriding requirement to test blood for infectious diseases. The obligations in the range $1000 < x < 2000$ are the specific tests to be performed. They have been enumerated like this because a subset of them can be specifically annulled by other clauses further on in the document. The satisfaction clauses are in response to the two negative test results. Note that they are presented in case evidence changes in future, and the original obligations again become active.

Obligation 2000 is a requirement to use FDA approved tests, obligation 7000 is a requirement to test blood before shipping it onto other parties, and obligation 9000 is a further need to test blood donations for Syphilis.

In figure 3, the situation is presented very near completion of the requirements for FDA compliance on this particular blood donation. From the event view, it is clear that six screening tests have been performed (in whatever order suited 'agentx'), and all their results are negative. In addition the screening tests have been declared to be FDA compliant.

In the fluent pane, note that although obligation 1000 is now satisfied (through the collective satisfaction of its subclauses), obligation 7000 is still outstanding. This is because it is explicitly dependent on both obligation 1000 and obligation


750:821 - DocTag frameset

http://www.cl.cam.ac.uk/~dme26/ra07L/FDAvis/

Control panel | Reset | time: -1 | 1 | +1 | Subject item: donation(1)

Deontic fluents up to time: 8.

- **Satisfaction:** →1000 data: [donation(1)]
- **Satisfaction:** →1100 data: [donation(1)]
- **Satisfaction:** →1200 data: [donation(1)]
- **Satisfaction:** →1300 data: [donation(1)]
- **Satisfaction:** →1400 data: [donation(1)]
- **Satisfaction:** →1500 data: [donation(1)]
- **Satisfaction:** →1600 data: [donation(1)]
- **Satisfaction:** →2000 data: [donation(1)]
- **Obligation:** →7000 data: [donation(1)]
- **Obligation:** →9000 data: [donation(1)]

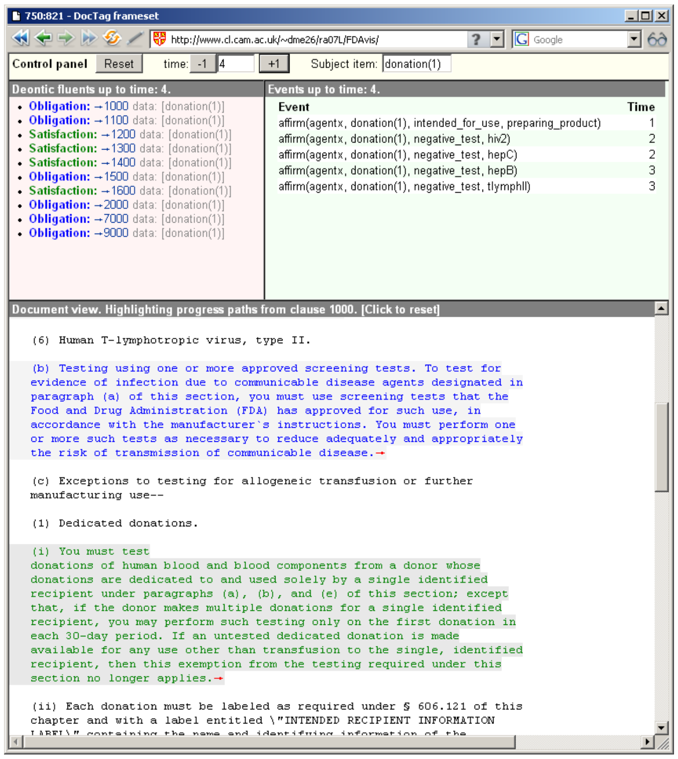
Events up to time: 8.

Event	Time
affirm(agentx, donation(1), intended_for_use, preparing_product)	1
affirm(agentx, donation(1), negative_test, hiv2)	2
affirm(agentx, donation(1), negative_test, hepC)	2
affirm(agentx, donation(1), negative_test, hepB)	3
affirm(agentx, donation(1), negative_test, tlymphI)	4
affirm(agentx, donation(1), negative_test, tlymphII)	3
affirm(agentx, donation(1), negative_test, hiv1)	5
affirm(agentx, donation(1), screening_tests_fda_compliant, true)	6

Document view.

```
acceptable for such purposes by FDA under § 610.41(b); and

(B) tests performed under paragraphs (a) and (b) of this section are
nonreactive.

(v) Anti-HBc reactive donations, otherwise nonreactive when tested as
required under this section, may be used for further manufacturing
into plasma derivatives without prior FDA approval or a \"BIOHAZARD\"
legend as required under paragraphs (h)(2)(ii)(A) and (h)(2)(ii)(B) of
this section.

(vi) You may use human blood or blood components, excluding Source
Plasma, that test reactive by a screening test for syphilis as
required under paragraph (i) of this section if, consistent with §
640.5 of this chapter, the donation is further tested by an adequate
and appropriate test which demonstrates that the reactive screening
test is a biological false positive. You must label the blood or blood
components with both test results.

(vii) You may use Source Plasma from a donor who tests reactive by a
screening test for syphilis as required under § 610.40(i) of this
chapter, if the donor meets the requirements of § 640.65(b)(2) of this
chapter.

(i) Syphilis testing. In addition to the testing otherwise required
under this section, you must test by a serological test for syphilis
under §§ 640.5(a), 640.14, 640.23(a), 640.33(a), 640.53(a), and
640.65(b)(2) of this chapter.→
```

Fig. 3. Screen-capture of document annotations for a completed workflow


750:821 - DocTag frameset

http://www.cl.cam.ac.uk/~dme26/ra07L/FDAvis/

Control panel | Reset | time: -1 | 4 | +1 | Subject item: donation(1)

Deontic fluents up to time: 4.

- **Obligation:** →1000 data: [donation(1)]
- **Obligation:** →1100 data: [donation(1)]
- **Satisfaction:** →1200 data: [donation(1)]
- **Satisfaction:** →1300 data: [donation(1)]
- **Satisfaction:** →1400 data: [donation(1)]
- **Obligation:** →1500 data: [donation(1)]
- **Satisfaction:** →1600 data: [donation(1)]
- **Obligation:** →2000 data: [donation(1)]
- **Obligation:** →7000 data: [donation(1)]
- **Obligation:** →9000 data: [donation(1)]

Events up to time: 4.

Event	Time
affirm(agentx, donation(1), intended_for_use, preparing_product)	1
affirm(agentx, donation(1), negative_test, hiv2)	2
affirm(agentx, donation(1), negative_test, hepC)	2
affirm(agentx, donation(1), negative_test, hepB)	3
affirm(agentx, donation(1), negative_test, tlymphII)	3

Document view. Highlighting progress paths from clause 1000. [Click to reset]

```
(6) Human T-lymphotropic virus, type II.

(b) Testing using one or more approved screening tests. To test for
evidence of infection due to communicable disease agents designated in
paragraph (a) of this section, you must use screening tests that the
Food and Drug Administration (FDA) has approved for such use, in
accordance with the manufacturer`s instructions. You must perform one
or more such tests as necessary to reduce adequately and appropriately
the risk of transmission of communicable disease.→

(c) Exceptions to testing for allogeneic transfusion or further
manufacturing use--

(1) Dedicated donations.

(i) You must test
donations of human blood and blood components from a donor whose
donations are dedicated to and used solely by a single identified
recipient under paragraphs (a), (b), and (e) of this section; except
that, if the donor makes multiple donations for a single identified
recipient, you may perform such testing only on the first donation in
each 30-day period. If an untested dedicated donation is made
available for any use other than transfusion to the single, identified
recipient, then this exemption from the testing required under this
section no longer applies.→

(ii) Each donation must be labeled as required under § 606.121 of this
chapter and with a label entitled \"INTENDED RECIPIENT INFORMATION
```

Fig. 4. Screen-capture of progress indication

9000. The highlighting of obligations in this case serves to indicate that related requirements (i.e. particular tests) are not necessarily close together in the policy document.

B. Compliant progress paths within a contract

There are likely to be many ways in which to progress the tasks implied by a policy document. As mentioned above, instead of encoding workflow, this paper suggests using the Event Calculus to determine likely paths of progress and indicate them on the source natural language policy document. Here the term 'progress' is used to mean satisfaction of obligations, or performing actions that annul violations.

Users of the system click the link directly following an obligation or violation clause to search for progress clauses connected with it. Because the policy document has been translated (in a "lossy" sense of the word!) into a logic program, it is reasonably straightforward to have the Prolog server reflect on its own code in the context of the current state of affairs to determine what clauses might perform the required annulments.

Because no specified proper subset of logic programming is currently required on the document representation, the safest answer for possible progress paths will be an over-estimation (i.e. clauses whose head can unify with the query source). In the CFR document encoded for this paper, a search that under-estimates the clause connections was chosen, because it was clear that a few manual annotations would suffice to complete the required relationship representation. Figure 4 shows a clause progress highlight generated from a progress query on the initial clause in the document.

Note that in many cases a clause will imply a method for its own satisfaction. For example, the obligation clause near the top of the document that requires users to test a blood donation for HIV type 1, can be satisfied by asserting that an HIV type 1 screening test has been performed. The document progress indicator does not search for this reflexive type of satisfaction of an obligation.

In the particular case shown in figure 4, the highlighted progress clause describes one of the exceptions to the requirement to perform screening tests on a blood donation. In broad terms, the highlighted clause annuls the requirement for testing when there is a single named recipient, and sufficiently recent tests have been performed on donations from the same donor.

C. Document graph

Given a software representation of the document that indicates how clauses relate to each other, it is possible to generate a graph representation of way the policy clauses interrelate in the original policy document. Figure 5 shows a graphical representation of the policy document with nodes of interest listed in nodes that are ordered top-to-bottom within left-to-right columns. The nodes represent the source and destination clauses of deontic relationships indicated by the edges between them. An edge from clause 'A' to clause 'B' indicates that clause 'A' may be able to annul clause 'B'.

Knowing that the document clause identifiers are fairly evenly spaced throughout the policy document, figure 5 indicates that while local collections of related clauses are

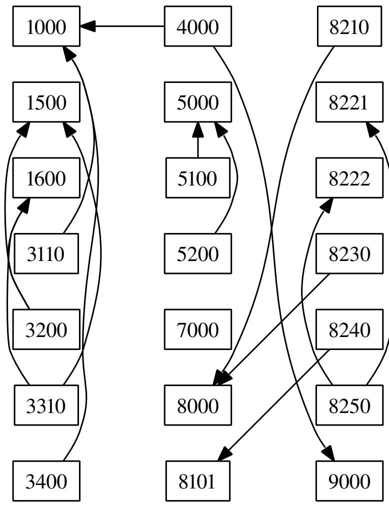


Fig. 5. Document clause interaction graph

common, there are numerous annulments (and thus clause interrelationships) that span considerable distances within the document. Within the figure, clause 8000 is the obligation not to ship reactive blood donations. Clause 4000 is not nearby lexically, yet indicates the annulment of the need to test blood donations being kept for future use targeted back at the same donor. Satisfying clause 4000, by declaring a donation autologous and satisfying various other obligation subclauses, will annul obligation clause 8000.

In future, it is intended that spacial document presentations, such as the one presented here, be used to provide users with a high-level user interface for navigating within very large policy documents.

VI. Future work

There is a significant amount of future research stemming from this project. Beyond experimentation with further documents, user testing will be of interest as the software matures.

Of particular interest is increasing the support for annotating the natural language documents and encoding the logic programs that represent them. Numerous predicate patterns emerged when encoding the FDA clauses although it remains an open question whether employing these predicate patterns will represent a useful number of other policy documents. The logical (but currently impractical) extent of this automation would be to use natural language processing techniques to convert policy documents to logic programs automatically.

As discussed in section V, the current prototype does not provide comprehensive facilities to manage the event list. Providing a user interface to manage event lists would clearly be useful to support the deductive processes demonstrated so far. However, the Event Calculus also facilitates other types of inference. Applying its inductive reasoning capabilities would help searching for sequences of user actions that reach desired policy states. Abductive reasoning with the Event Calculus could be used to determine where source documents, or their encoding in software, appear to be deficient.

Even without more extensive inference mechanisms, the Event Calculus could be directly useful in performing what-if analyses over sequences of hypothetical events.

VII. Conclusion

This paper has presented a user interface for policy compliance testing and progress assistance that focuses on annotation of natural language policy documents, with a parallel logic program representation of their content. A section of the US FDA CFR relating to blood testing was encoded, and the prototype implementation of the system demonstrated to provide appropriate guidance to potential users. In contrast to prescriptively encoded Petri Nets, Finite State Machines or other workflow graph approaches, the Event Calculus deontic state monitoring and associated user interface allow users to comply with policy in a manner that suits them.

References

[1] J. Wielemaker, “SWI Prolog,” http://www.swi-prolog.org/, 1987.
[2] M.-T. Schmidt, “The evolution of workflow standards,” *IEEE Concurrency*, 1999.
[3] R. W. Bons, R. M. Lee, R. W. Wagenaar, and C. D. Wrigley, “Modelling inter-organizational trade procedures using documentary petri nets,” in *Proceedings of the Hawaii International Conference on System Sciences*, 1995.
[4] A. Daskalopulu, “Logic-based tools for the analysis and representation of legal contracts,” Ph.D. dissertation, Department of Computing, Imperial College, University of London, 1999.
[5] R. M. Lee, “Bureaucracies as deontic systems,” *ACM Transactions on Office Information Systems*, vol. 6, no. 2, pp. 87–108, Apr. 1988.
[6] A. Daskalopulu, T. Dimitrakos, and T. S. Maibaum, “E-contract fulfillment and agents' attitudes,” in *Proceedings ERCIM WG E-Commerce Workshop on the Role of Trust in E-Business*, Zurich, Oct. 2001.
[7] B. N. Grosof, Y. Labrou, and H. Y. Chan, “A declarative approach to business rules in contracts: Courteous logic programs in XML,” in *Proceedings First ACM Conference on Electronic Commerce (EC-99)*, M. P. Wellman, Ed., Nov. 1999.
[8] A. S. Abrahams, “Developing and executing electronic commerce applications with occurrences,” Ph.D. dissertation, University of Cambridge Computer Laboratory, 2002.
[9] A. S. Abrahams, D. M. Eyers, and J. M. Bacon, “Practical contract storage, checking, and enforcement for business process automation,” in *Formal Modeling for Electronic Commerce: Representation, Inference, and Strategic Interaction*, S. O. Kimbrough and D. Wu, Eds. Springer-Verlag, 2004, pp. 33–77.
[10] ——, “Regulating web-based communities,” in *In Proceedings of the IADIS International Conference on Web Based Communities (WBC2004)*, Lisbon, Portugal, Mar. 2004.
[11] J. J. Meyer and R. J. Wieringa, *Deontic Logic in Computer Science*. John Wiley & Sons Ltd, 1993.
[12] R.Kowalski and M.Sergot, “A logic-based calculus of events,” *New Generation Computing*, vol. 4, pp. 67–95, 1986.
[13] M. Shanahan, “The event calculus explained,” *Springer Lecture Notes in Artificial Intelligence*, vol. 1660, pp. 409–30, 1999.
[14] N. Dinesh, A. Easwaran, D. Arney, A. Abrahams, O. Rambow, A. Joshi, and I. Lee, “Extracting traceable formal models from natural language policy,” Poster presented at the annual research review and workshop on High-Confidence Embedded Systems, Lincoln, Nebraska, 2005. [Online]. Available: http://www.cis.upenn.edu/~nikhild/Papers/poster.pdf
[15] N. Dinesh, A. Joshi, I. Lee, and B. Webber, “Extracting formal specifications from natural language regulatory documents,” in *Proceedings of the Fifth International Workshop on Inference in Computational Semantics (ICoS-5)*, Buxton, England, 2006. [Online]. Available: http://www.cis.upenn.edu/~nikhild/Papers/specifications_icos.pdf
[16] N. Freed and N. Borenstein, “Multipurpose internet mail extensions (MIME) part one: Format of internet message bodies,” 1996, rFC 2045.