Integrating Discourse and Domain Knowledge for Document Drafting

L. Karl Branting
Department of Computer Science
University of Wyoming
P.O. Box 3682
Laramie, WY 82071
USA
karl@uwyo.edu

Charles B. Callaway, Bradford W. Mott, James C. Lester
Department of Computer Science
North Carolina State University
Box 8206
Raleigh, NC 27606
USA
{cbcallaw,bwmott,lester}@eos.ncsu.edu

Abstract

Document drafting is a key component of legal expertise. Effective legal document drafting requires knowledge both of legal domain knowledge and of the structure of legal discourse. Automating the task of legal document drafting therefore requires explicit representation of both these types of knowledge. This paper proposes an architecture that integrates these two disparate knowledge sources in a modular architecture under which representation and control are optimized for each task. This architecture is being implemented in DocuPlanner 2.0, a system for interactive document drafting.

1 Introduction

There is a growing recognition in the AI and Law community of the central role of document drafting in legal expertise. Lawyers engage in a wide range of activities involving legal expertise. Some of these activities have as their explicit goal the creation of documents intended to effect changes in legal status, such as drafting wills, contracts, leases, and deeds. However, even activities that do not have documents as their ultimate object, such as counseling clients, presenting arguments in courts and other forums, and negotiation, nevertheless entail drafting documents such as briefs, letters, and memoranda. Moreover, judicial decision-making entails creation of a wide range of documents, from brief orders to lengthy decisions [Bra98].

In general, the focus of legal knowledge-based system development has been on legal reasoning at the expense of knowledge concerning document drafting. The primary tasks addressed by most legal knowledge-based systems have been analysis, argumentation, prediction, and planning. While some systems, such as Hypo [Ash90] and Grebe [Bra91] produce memoranda as output, these memoranda are not intended to duplicate actual legal discourse as expressed in legal documents, and many other systems provide even more elliptical output. Conversely, the numerous commercial document drafting systems now available are, in general, strikingly devoid of explicit legal expertise [SL97].

This paper argues that automation of legal expertise requires explicit representation of both legal domain knowledge and knowledge of the structure of legal discourse. This is because legal reasoning and legal document drafting do not occur in isolation, but are instead mutually dependent components of legal expertise. However, there are significant differences in the representation and control requirements of legal reasoning and document drafting. Developing a model that encompasses both types of legal expertise therefore requires an architecture that integrates two sets of disparate techniques. This paper argues for a modular architecture under which representation and control can be optimized for each task.

Section 2 sets forth several key desiderata for document drafting systems. Section 3 argues for both the importance and the feasibility of using explicit discourse knowledge in legal document drafting. The role of explicit domain knowledge is discussed in Section 4. Our architecture for achieving this integration is set forth in Section 5.

2 Desiderata for Document Drafting Systems

Document creation is a synthesis task in which the user posits a set of communicative and pragmatic goals, creates a rhetorical structure for the evolving document, and constructs texts that achieve the goals. The purpose of document creation systems is to reduce the burden on the drafter to the greatest possible extent by automating document synthesis. Ideally, a document creation system should enable the user to provide the facts of the current case through user-initiated processes (*e.g.*, filling in forms), a system-initiated interview, or some mixture of the two. It should then generate the entire document, which would then be presented to the drafter, perhaps for some minimal post-editing.

Currently, relatively simple techniques can be employed to draft documents for document classes whose rhetorical structures vary little across instances [Lau92]. For example, petitions for restraining orders or filings for extensions of time require only the substitution of text strings within boilerplate, and template-based document creation systems can easily be constructed for tasks as straightforward as these. By instantiating a generic template with user-provided strings, such as for the names of parties involved in the case, an *unknowledgeable* document assembly system can easily produce the required documents.

However, there are many document classes for which these simple techniques are inadequate, *i.e.*, document classes that require significant legal reasoning and whose rhetorical structures therefore vary more widely across document instances. For example, drafting administrative or trial decisions and orders entails non-trivial legal reasoning. Moreover, the rhetorical structures of documents that achieve complex legal goals must somehow reflect the underlying domain reasoning. However, the precise relation between this reasoning process and the rhetorical structure of the document can be quite complex. Although some aspects of the inference procedure result in straightforward production of text in the document, the mapping of inference trees onto a document surface structure is frequently not at all straightforward.

Lauritsen has suggested that effective automated document creation systems should provide users with a variety of inspection and editing functionalities [Lau93]. These include tools for noting essential and optional document segments, browsing documents, validating drafts, and highlighting differences between multiple versions. While these functionalities are important, we believe that the most critical requirement for practical document creation systems is queryable liveness:

> *Queryable Liveness:* At any time, the drafter can inspect, modify, and pose questions about specific segments of the document to determine how that region of text follows from the case facts.

Results from key projects in the expert systems community suggest that explanation is a key functionality for successful deployment and adaptation for everyday practice [Swa83]. Users are typically unwilling to accept the conclusions drawn by expert systems unless the systems are able to justify their reasoning. Similarly, we believe that for many classes of documents, particularly those that are the product of relatively complex legal reasoning, attorneys will be much more inclined to use automatically drafted documents if they understand the origin of the documents' text. In response to justification queries, document drafting systems should therefore be able to provide explanations about the legal reasoning underlying the documents that they generate.

In summary, an effective document-drafting system should elicit in a concise and intuitive fashion the information necessary to create a document that satisfies the user's goals and should be capable of explaining the relationship between text regions and case facts in terms that are understandable to the user. The next section explains the role that discourse knowledge plays in these processes.

3 Discourse-Based Techniques for Document Drafting

Three main approaches to automated document drafting can be distinguished. The first is a *procedural* approach under which commands in an imperative programming language select text elements for combination based on user-provided, document-specific data. The second approach is *template-based*. Under this approach, a class of documents is represented by a template consisting of text common to all members of the class. Embedded in this text are (1) tokens representing document-specific facts and (2) tests (*e.g.*, IF-THEN statements) specifying the conditions under which text elements will be included in a particular document.

The third approach, termed the *discourse-based* approach, uses an explicit model of the discourse structure of classes of documents to guide creation of new documents. Discourse structure consists of the relationships between statements in a multi-sentential text that are responsible for the text's coherence. The roots of this approach are in speech-act theory, which is the study of illocutionary content of discourse, *i.e.*, the goals that speakers seek to accomplish through their discourse [Gri75, Sea69]. The relevance of speech-act theory to the AI and law community was first demonstrated by Anne Gardner in her system for contract formation [Gar87]. In the computational linguistics community, the insights of speech-act theory were used to develop techniques for understanding multi-sentential text by inferring the text's underlying discourse structure [MT87, GS86, Hob79]. This discourse structure comprises a number of rhetorical relations among sentences, such as elaboration, exemplification, generalization, and sequence. In contrast to their role in natural language understanding, discourse relations can be used "in reverse" in natural language generation to dynamically construct coherent text.

In our previous work [BLC98, BLC97, BL96] we proposed a discourse model consisting of two elements: (1) an illocutionary goal structure that expresses the goal dependencies among the relevant legal predicates and the connection between performative text segments and the illocutionary goals that they achieve; and (2) a rhetorical structure that expresses the stylistic and discourse conventions of the document's genre. The illocutionary and rhetorical structures of a document together constitute the document's *discourse structure*. In [BLC97] we showed how the discourse structure of documents created with a *discourse grammar* can be used to answer questions about why text segments were included and how propositions expressed by text segments are justified, and in [BLC98] we showed how a discourse grammar can be used to generate a wide range of documents within a particular document class.

3.1 Limitations of Current Commercial Systems

Current commercial legal document drafting systems rely almost exclusively on "knowledge-free" procedural or template-based approaches. Lauritsen's recent survey [SL97] reveals two important limitations of these current-generation document technologies. First, none of the systems exploit sophisticated representations of linguistic knowledge. For example, the most advanced form of linguistic knowledge encoded by these systems appears to be knowledge about the conditional inclusion of text segments, *e.g.*, particular clauses in a contract. In contrast to significantly more sophisticated models of document assembly advocated by AI researchers [RML95, Lau93, Gor89], they are therefore unable to reason about the rhetorical structure and linguistic form of the documents they produce.

Second, none of the systems exploit representations of domain knowledge that has any of the sophisticated inference capabilities advocated by the legal reasoning community. For example, the ability to reason with knowledge about dates is considered an unusually advanced feature. Because these systems lack both linguistic and domain knowledge, they lack the flexibility that is necessary for dealing with the broad range of situations typical of many document creation tasks.

3.2 What NLG Can Bring to Document Drafting

Document generation has much to gain from lessons learned in natural language generation (NLG), an active area of research in computational linguistics for the past two decades. Concerned with the automatic generation of text from formal representations of knowledge, NLG is typically decomposed into two primary tasks: planning and realization [McK82].

Planning consists of determining the content and organization of multi-sentential discourse, and realization consists of translating formal representations of the selected knowledge into grammatically correct text. NLG differs significantly from the template-based approaches employed in most document assembly systems in its exploitation of linguistic knowledge. In much the same manner that models of legal reasoning can be used by expert systems to draw inferences leading to the solution to novel problems of jurisprudence, models of linguistic reasoning can be used by document systems to draw inferences leading to the creation of customized documents.

While template-based approaches either use implicit linguistic knowledge or remain agnostic about linguistic commitments in general, NLG systems employ a number of forms of linguistic knowledge to make decisions about the content, structure, and surface form of generated texts. First, they employ discourse knowledge to determine the rhetorical organization of propositions to be communicated. For example, the Rhetorical Structure Planner [Hov93] and Moore's explanation system [Moo95] use plan-based representations of discourse knowledge to impose a rhetorical organization on the multi-sentential texts they generate. Second, NLG systems employ sentence planning knowledge to make decisions about which propositions should be aggregated together and about how domain concepts should be referenced. For example, Dale's EPICURE system invokes a sophisticated referring-expression planner to determine how to pronominalize and how to create definite descriptions [Dal92]. Generators also use sentence planning knowledge to invoke aggregation techniques for creating relative clause embeddings and ellipsis. Third, NLG systems use lexical knowledge to make word planning decisions. Given a concept or relation, lexicalization entails selecting a word or phrase that properly expresses that concept or relation. This process may involve taking discourse context into account to select phrasings whose connotations are as carefully considered as their denotations. Fourth, NLG systems employ grammatical knowledge to transform sentence plans, which are typically expressed in a formal semantic representation, into grammatically correct sentences. For the past decade, this "realization" work has focused on systemic-functional grammars [Elh91] that consist of enormous English grammars, which take care of word order, agreement, and the insertion of functional words. Finally, NLG systems employ morphological knowledge to make decisions about the formation of words themselves. Morphology systems typically handle all matters relating to person, number, pronoun formation, and gender.

Applied natural language generation has been the subject of increasing attention in recent years, in large part because of the significant strides that have been made in more theoretically oriented NLG [RD97]. Reiter has argued that applied NLG systems' linguistic knowledge accrues important advantages over template-based approaches [Rei95]. First, systems with explicit NLG knowledge are more maintainable. Because they encode declarative representations of linguistic knowledge about how to determine the content, structure, and form of generated texts, these systems can be updated to reflect knowledge changes without extensive revision. Second, Reiter argues that they can create higher-quality texts than templates. Because they reason about how to plan multi-sentential texts, the content of individual sentences, and the grammatical properties of words and phrases, they can (in principle at least) create texts whose clarity and coherence is greater than template-based approaches. Implicit in this observation is the fact that applied NLG systems are intended to handle fairly wide ranging inputs. Applied NLG systems are therefore much less subject to the difficulty of precisely anticipating the set of variable instantiations and conditional texts required for a document class than are template-based systems. Third, Reiter argues that applied NLG systems permit documents to be generated that more easily provide multi-lingual output and conform to specified textual standards.

While it is true that maintainability, higher quality texts, and multi-linguistic output are important, we believe that the overwhelming advantage of natural language generation for document creation is the flexibility it offers for dealing with the idiosyncratic domain-specific inferences involved in the creation of documents. This flexibility is essential for providing the functionalities outlined in Section 2. In particular, without the ability to dynamically plan the rhetorical structure of a document, it is quite difficult to create documents to satisfy a set of goals that are in fact themselves highly dependent on domain-specific inferences to be drawn about the specifics of the input "case" and are not determined until runtime.

Applied NLG systems offer a knowledgeable alternative to traditional approaches to document creation. Most applied NLG systems offer some subset of the above functionalities. In general, knowledge that resides higher in the NLG pipeline occurs more frequently in applied NLG systems than knowledge that is more grammar-oriented. It has been hypothesized that, of all the functionalities of NLG systems, discourse planning—rhetorical planning typically includes both content selection and rhetorical structuring in multi-sentential texts—may provide the strongest value-added [Rei95]. While sentence planning (aggregation, referring expression planning, and lexical choice) and realization (creating the syntactic form of individual sentences) may also confer benefits to document creation systems, discourse planning currently appears the most promising.

Discourse planning requires precisely the kind of linguistic knowledge that is readily available from the analysis of document corpora. By inspecting representative samples of the class of documents to be created, one can abstract a generic set of rhetorical operators to provide comprehensive or near-comprehensive coverage. Furthermore, discourse planning does not require extensive sentence planning knowledge or grammatical knowledge, both of which would require significant investments in developing representation formalisms, creating systems to reason with them—off-the-shelf sentence planners, for example, are not currently available—and maintaining them as surface structures of the documents evolved. By adopting a hybrid approach that employs the applied NLG techniques of discourse planning and simple (less sophisticated but less labor intensive) approaches to sentence planning and realization, a document creation system can enjoy the benefits of applied NLG while avoiding much of the associated overhead.

3.3 NLG and Legal Documents

Two factors peculiar to legal documents make NLG techniques particularly appropriate for them. The first concerns the feasibility of NLG techniques and the second concerns their necessity. First, many legal documents have an extremely well-defined discourse structure arising from subgoal relations between legal predicates. This subgoal structure is the basis of models of legal discourse derived from Toulmin's model of argument structure, which analyzed argumentative texts in terms of the concepts of warrant, ground, conclusion, backing, and qualification [Tou58]. Toulmin's model has been used for explanation generation [ZS95], document drafting in PLAID [BCS95], and analysis [FF96]. Goal trees were also the basis of the document models of JEDA [PRK89] and LAWCLERK [Bra93]. Performative documents, such as orders, judgments, wills, and contracts, and argumentative documents, such as briefs, generally consist in part of text segments clearly connected to specific legal predicates. Thus, the discourse structure needed for NLG techniques is particularly clear and unambiguous in legal documents.

The second factor is that text segments associated with a given legal predicate are, in general, interleaved with text associated with other legal predicates. In judicial orders, for example, assignment of a value of "true" to a predicate may require separate, noncontiguous text segments in the "finding," "ruling," and "order" sections of the order. For documents with this *interleaved text property* the final document text cannot in principle be generated by concatenating text encountered during a recursive traversal of the inference tree. *A fortiori*, simpler template-based approaches that lack even explicit representations of legal reasoning in the form of an inference tree are extremely ill-suited for such documents.

Various ad hoc approaches to mapping inference trees to document text have been proposed, *e.g.*, the mapping strategies set forth in [Bra93]. However, a principled approach to documents with the interleaved text property requires the explicit representation of ordering constraints provided by applied NLG techniques.

In summary, discourse knowledge is essential for the flexibility to generate any but the simplest legal documents.

4 Domain Expertise for Document Drafting

Discourse knowledge is essential for flexible and accurate drafting of complex legal documents. Discourse knowledge is not, however, sufficient *per se* for documents that embody significant legal expertise. Such documents also require explicit legal domain knowledge.

The illocutionary structure of legal documents consists of two elements: an inference tree expressing subgoal relations among the applicable legal predicates; and the connection between performative text segments and the illocutionary goals that they achieve. The former is precisely the goal tree generated (explicitly or implicitly) by a rule-based or logic-based reasoning system. Accordingly, if a document-drafting system requires an explicit representation of illocutionary structure, as is required by the discourse-based NLG techniques discussed above, then an inference tree must be generated in the process of document generation.

Effective creation of an inference tree in turn requires both a declarative representation of underlying legal rules and a control strategy to guide acquisition of case facts and inferences. Many of the familiar criteria for rule-based expert systems (see, *e.g.*, [BS84]) are applicable to illocutionary tree creation:

- Users should be asked only the minimum number of questions necessary to elicit the facts necessary for the decision.

- The questions should be expressed in idiomatic natural language.

- The questions should be ordered in a fashion that corresponds to the manner that users think about the domain.

- The system should be capable of explaining why it is asking a question and how it reached a conclusion.

- The system should be capable of mixed-initiative data acquisition; the user should have the option of providing some data by filling in forms, rather than being asked about every data item.

- Search strategies should depend on the nature of the particular goals, *e.g.*, satisficing for date calculations, exhaustive for jurisdictional defects (since orders of dismissal often set forth all jurisdictional defects).

Thus, legal domain knowledge is essential for effective document drafting for two reasons. First, the top-level illocutionary structure of many legal documents consists of an inference tree. Functions that depend on the illocutionary structure, such as the explanation capability necessary for document "queryable liveness," therefore depend on the legal domain knowledge necessary to generate such inference trees. Second, effective case fact acquisition requires a declarative representation of the underlying legal rules together with knowledge about how experts use and think about those rules. Thus, effective and flexible document drafting requires both discourse knowledge and legal expertise.

The importance of both types of knowledge is illustrated by our experience with DOCUPLANNER 1.0 [BLC98]. Given a set of case facts, DOCUPLANNER 1.0 planned the rhetorical structure of the documents and from this structure created the documents themselves. While DOCUPLANNER 1.0 constituted an advance over legal document drafting systems without linguistic knowledge, it nonetheless suffered from an inadequate treatment of domain knowledge. It employed a document grammar that represented not only discourse knowledge, but also domain knowledge with which it drew inferences about the given case facts to make findings. This conflation of linguistic knowledge and discourse knowledge proved to be troublesome because it required the unification-based control mechanism of the document planner also to be used for domain reasoning.

Unfortunately, while discourse planning can proceed in a rather straightforward top-down manner, drawing inferences to make findings needs to be a highly interactive process involving significant give-and-take with the user. Furthermore, it often needs to invoke specialized computational mechanisms, *e.g.*, drawing inferences about dates. None of these requirements is easily accommodated by a document grammar.

Hence, attempts to subsume legal inference mechanisms under a document planning mechanism are likely to meet with limited success. We therefore argue for a model of document generation that (1) provides both domain knowledge and discourse knowledge, and (2) cleanly separates these kinds of knowledge and their respective inference mechanisms in a modular architecture.

5 DocuPlanner 2.0: A Modular Architecture

To address the design criteria set forth above, we have formulated a document planning architecture that employs both domain and discourse knowledge in a modular fashion. The document creation model (Figure 1) relies on four primary

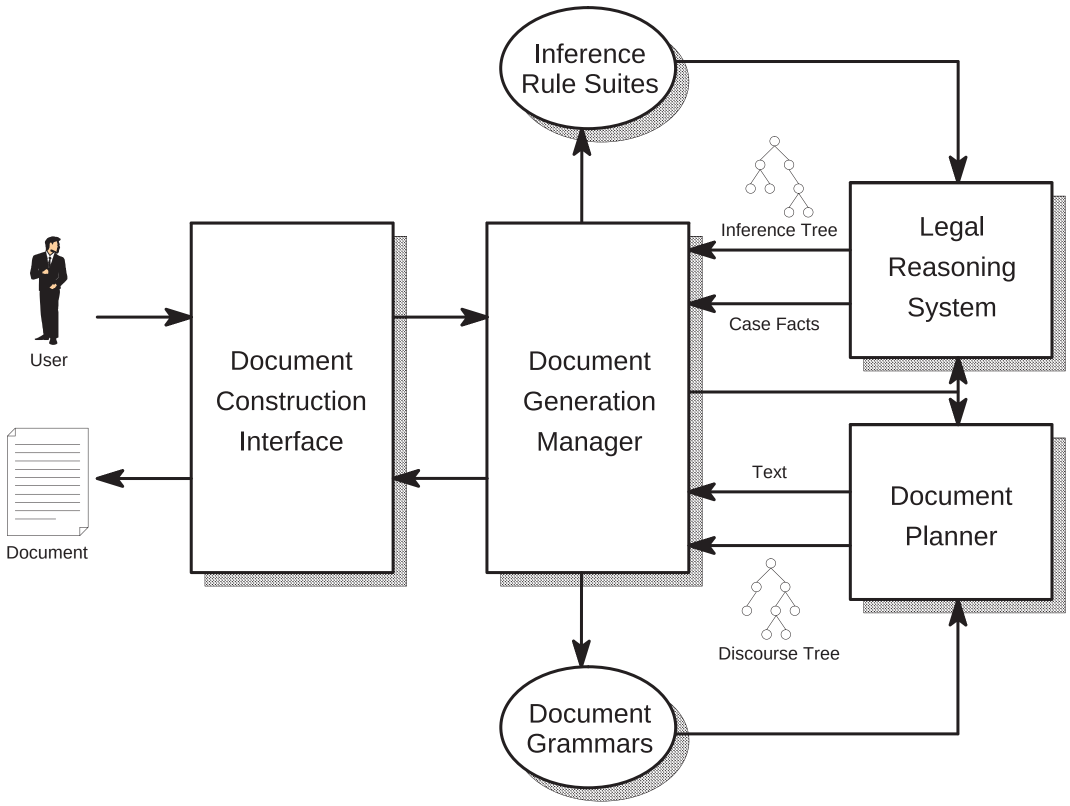

Figure 1: The architecture of DocuPlanner 2.0.

components: a *document generation manager*, a *document construction interface*, a *legal reasoning system*, and a *document planner*. Implementation of each of these components is currently underway in our laboratories. Below we sketch their operation.

When a user wishes to draft a new document, she indicates the class of documents, *e.g.*, show-cause orders, to which the new document should belong. This request is passed to the document generation manager, which loads (1) an inference rule suite and (2) a document grammar, both designed specifically for creating documents of that class. The inference rule suite is loaded into the legal reasoning system and the document grammar is loaded into the document planner. The legal reasoning system begins with the top-most legal predicate and subgoals on it. As it encounters primitive goals, it poses questions to the user to ascertain their values.

Primitive goals are marked as being unitary or collective. Unitary goals cause questions to be posed immediately; collective goals cause multiple questions to be asked of the user at one time. This mechanism enables the system to compose aggregate dialogue boxes that avoid the irritation sometimes experienced by users when subjected to a long series of queries by the system. Users respond to these questions with text strings and by indicating their choices with GUI selections. At any time, users may ask why they are being queried for the requested information, and the system will provide an explanation of its line of interrogation.

Subgoaling continues until the top-most predicate has been established, *i.e.*, all information associated with primitive subgoals has been provided. Where specified by the inference rule suite, the legal reasoner invokes specialized computational mechanisms that are idiosyncratic to that class of documents. The net result of the legal reasoner's efforts is a fully instantiated inference tree and a set of case facts, which are returned to the document generation manager.

Next, the document planner inspects the instantiated inference tree created by the legal reasoner. Its task is to interpret the findings of the legal reasoner in the context of the appropriate document grammar to create a well-formed document of the specified class. To do so, it employs a two-phase process in which document planning is followed by document drafting. First, during document planning, the planner constructs a fully instantiated discourse structure [Hov93] for the evolving document. To construct the discourse structure, the planner employs a unification-based reasoner that takes the findings and case facts as input and unifies them with the document grammar.

Each leaf of the discourse structure consists of a small number of propositions to be communicated, a sentence skeleton for communicating them, a set of bindings of case facts to sentence skeleton variables, and (optionally) formatting directives. The discourse structure also includes a set of global formatting directives which are subsequently used in the subsequent drafting phase. Next, during drafting, the planner conducts a left-to-right traversal of the leaves of the discourse structure to form a linear text. At each leaf, it (1) instantiates the sentence skeleton(s) with the case fact bindings, (2) invokes a morphological system to create words with correct person, number, and gender, (3) invokes an orthographic system to construct necessary punctuation, and

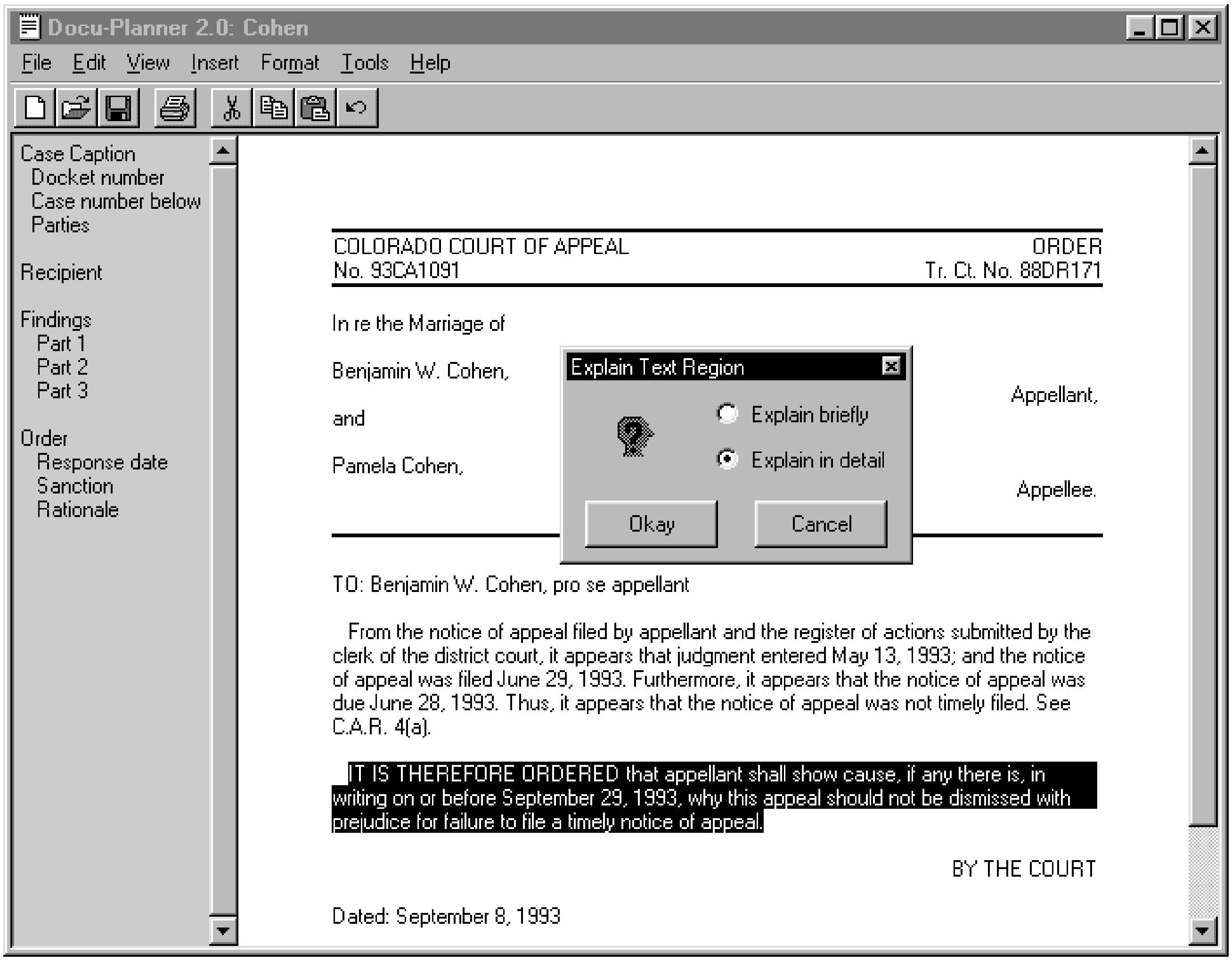

Figure 2: A sample screen from DocuPlanner 2.0.

(4) applies the specified formatting directives. Finally, the drafting phase is completed by concatenating the text obtained from the traversal to obtain the document. The planner then passes the fully instantiated discourse plan and the document itself to the document generation manager, which then instructs the interface to present it to the user.

At this juncture the user now has a number of options. She may browse the document in one of two ways: by scrolling through it, or by using the outline navigation tool, which displays headings and subheadings generated by internal nodes in the instantiated discourse structure. While inspecting the document, she may realize that one or more of the case facts has been entered erroneously. By invoking the case facts editor she can modify the entries as she sees fit. Then, by selecting *regenerate*, she can request the system take the modifications into account as it creates a revised draft.

To address the "queryable liveness" requirement, the interface provides the user with the ability to highlight a region of text and request an explanation of why that segment was included in the document (Figure 2). An explanation request triggers the following series of events. First, the interface notes the region of text specified by the user and inspects the discourse tree to determine the most specific (lowest) interior node of the instantiated discourse structure to which the user is referring. It passes this information to the document generation manager, which (1) inspects justification annotations on the corresponding regions of the discourse tree, (2) traverses regions of the inference tree referenced by the selected discourse tree regions, and the discourse tree, both of which are instantiated with case facts, and (3) generates an explanation that links the case facts, the findings of the legal reasoner, and the rhetorical structure specified by the current document class. The user may request either terse or verbose explanations and is free to pose follow-up requests to obtain more detail. She may then modify the case facts and regenerate the document until she is satisfied with its form and content. Finally, she can post-edit the document into its final form.

This architecture is being implemented in DocuPlanner 2.0, a document creation system for judicial document assembly. DocuPlanner 2.0's legal reasoner builds on our previous work on JSA [Bra98], a Lisp implementation of a back-chaining reasoner for judicial screening. DocuPlanner 2.0's document planner is being implemented in Fuf [Elh92], a unification system developed at Columbia that we have been using in document planning projects over the past several years.

6 Conclusion

Because of the complexities inherent in legal reasoning and composition, document drafting presents serious challenges to the AI and Law community. Not only must document systems be able to semi-automatically create a broad range of documents, they must be able to explain the reasoning that underlies the content and structure of the documents as well. Especially problematic is the fact that the structure of the inference tree produced during legal reasoning is typically *not* isomorphic to the rhetorical tree that will define the organization of the resulting document.

To address these challenges, this paper has proposed a model of document creation that exploits both domain knowledge and discourse knowledge in a modular architecture. Domain knowledge consists of inference rules, while discourse knowledge consists of document grammars that encode rhetorical structure relationships. Domain knowledge is used by a legal reasoning system to interrogate the user and makes findings by drawing inferences about user-supplied case facts. Discourse knowledge is used by the document planner to construct a discourse tree that defines the organization of the document. This model is being implemented in a document creation system for generating judiciary documents whose content and structure can be queried by the user and dynamically explained. A particularly challenging area of work lies in developing a unified representational formalism that is amenable to both the generation of fluent text for documents and clear explanations for document creators. We will be exploring these issues in our future work as the implementation progresses.

Acknowledgments

This research was supported in part by the North Carolina State University IntelliMedia Initiative, the University of Kaiserslautern Center for Learning Systems and Applications, the German-American Fulbright Commission, and a University of Wyoming Flittie sabbatical award.

References

[Ash90] K. Ashley. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals.* MIT Press, Cambridge, Massachusetts, 1990.

[BCS95] T. Bench-Capon and G. Staniford. PLAID - proactive legal assistance. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 81–88, 1995.

[BL96] L.K. Branting and J. C. Lester. A framework for self-explaining legal documents. In *Proceedings of the Ninth International Conference on Legal Knowledge-Based Systems (JURIX-96)*, pages 77–90, Tilburg University, the Netherlands, December 1996.

[BLC97] K. Branting, J. Lester, and C. Callaway. Automated drafting of self-explaining documents. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law (ICAIL-97)*, pages 72–81, University of Melbourne, Melbourne, Australia, June 30–July 3 1997. ACM Press.

[BLC98] L. Branting, J. Lester, and C. Callaway. Automating judicial document drafting: A unification-based approach. *Artificial Intelligence and Law*, 6(2–4):111–149, 1998.

[Bra91] K. Branting. Building explanations from rules and structured cases. *International Journal of Man-Machine Studies*, 34:797–837, 1991.

[Bra93] K. Branting. An issue-oriented approach to judicial document assembly. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*, pages 228–235, Amsterdam, The Netherlands, June 15–18, 1993. ACM Press.

[Bra98] L. Karl Branting. Techniques for automated drafting of judicial documents. *International Journal of Law and Information Technology*, 6(2):214–229, 1998.

[BS84] B. Buchanan and E. Shortliffe. *Rule-Based Expert Systems.* Addison-Wesley Publishing Co., Menlo Park, 1984.

[Dal92] Robert Dale. *Generating Referring Expressions.* MIT Press, 1992.

[Elh91] M. Elhadad. FUF: The universal unifier user manual version 5.0. Technical Report CUCS-038-91, Department of Computer Science, Columbia University, 1991.

[Elh92] M. Elhadad. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation.* PhD thesis, Columbia University, 1992.

[FF96] K. Freeman and A. Farley. A model of argumentation and its application to legal reasoning. *Artificial Intelligence and Law*, 4(3–4):163–197, 1996.

[Gar87] A. Gardner. *An Artificial Intelligence Approach to Legal Reasoning.* Bradford Books/MIT Press, Cambridge, MA, 1987.

[Gor89] T. Gordon. A theory construction approach to legal document assembly. In *Pre-Proceedings of the Third International Conference on Logic, Informatics, and Law*, pages 485–498, Florence, 1989.

[Gri75] H. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics 2: Speech Acts*, pages 41–58. Academic Press, New York, N.Y., 1975.

[GS86] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[Hob79] J. Hobbs. Coherence and co-reference. *Cognitive Science*, 3(1):67–82, 1979.

[Hov93] E. H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385, 1993.

[Lau92] M. Lauritsen. Technology report: Building legal practice systems with today's commercial authoring tools. *Law and Artificial Intelligence*, 1(1), 1992.

[Lau93] M. Lauritsen. Knowing documents. In *Fourth International Conference on Artificial Intelligence and Law*, pages 185–191, Amsterdam, 1993. ACM Press.

[McK82] K. McKeown. *Generating Natural Language Text in Response to Questions about Database Structure.* PhD thesis, University of Pennsylvania, 1982.

[Moo95] J. D. Moore. *Participating in Explanatory Dialogues.* MIT Press, 1995.

[MT87] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC/Information Sciences Institute, Marina del Rey, CA, June 1987.

[PRK89] V. P. Pethe, C. P. Rippey, and L. V. Kale. A specialized expert system for judicial decision support. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 190–194, Vancouver, B.C., June 13-16 1989.

[RD97] Ehud Reiter and Robert Dale. Building applied natural-language generation systems. *Journal of Natural-Language Engineering*, 3:57–87, 1997.

[Rei95] Ehud Reiter. NLG vs. templates. In *Proceedings of the Fifth European Workshop on Natural-Language Generation*, Leiden, The Netherlands, 1995.

[RML95] E. Reiter, C. Mellish, and J. Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9:259–287, 1995.

[Sea69] J. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge, 1969.

[SL97] A. Soudakoff and M. Lauritsen. Shopper's guide to legal document assembly. *Law Office Computing*, October/November 1997.

[Swa83] William R. Swartout. XPLAIN: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21:285–325, 1983.

[Tou58] S. E. Toulmin. *The Uses of Argument.* Cambridge University Press, 1958.

[ZS95] J. Zeleznikow and A. Stranieri. The Split-Up system: Integrating neural networks and rule-based reasoning in the legal domain. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 185–194, 1995.