Techniques for Automated Drafting of Judicial Documents

L. Karl Branting
Department of Computer Science
University of Wyoming
Laramie, WY
USA

Abstract

Document drafting is an essential component of judicial problem solving. This paper distinguishes several classes of judicial documents based on (1) the stage of the judicial process in which they are created, (2) the complexity of the documents, and (3) the party who drafts the documents: a judge, judicial support personnel, or a litigant. Three approaches to automated document drafting are identified and the applicability of these approaches to each class of judicial document is described. The paper concludes with a description of several implemented prototype systems for drafting judicial documents at various stages of the judicial process.

1 Introduction

Judicial reasoning is one of the most challenging and complex legal activities. Judicial decision-making may require assessing the credibility of witnesses, evaluating the probative weight of evidence, interpreting the meaning and intended effect of legal statutes and precedents, and balancing competing policies and principles. Judges are generally required to justify their resolution of these diverse factors in written documents. Document drafting is therefore an essential activity of the judiciary.

There is a very wide range of complexity of judicial documents. At the apex of complexity are appellate decisions. In broad outline, the form of appellate decisions is often quite regular: appellate decisions typically set forth a summary of the relevant facts of the case, enumerate the legal issues raised in arguments by counsel for each of the parties, enunciate the legal propositions supported by the relevant legal authorities, and state a decision that resolves the issues by applying the legal propositions to the facts of the case. However, the specific content of such decisions is extremely variable. Cases that are appealed are typically those that involve novel factual situations or legal issues. Analyzing and resolving such cases is a highly creative and challenging activity.

At the opposite extreme of complexity from appellate decisions are routine notices and orders. A single case may give rise to numerous documents at both the trial and appellate levels relating to pleadings, discovery, time extensions, motions for dismissal or summary judgment, or sanctions for violations of trial or appellate rules. These documents typically have considerable stylistic and substantive consistency.

Judicial documents of all levels of complexity have very high requirements for correctness and consistency. Typically, all judicial decisions and orders except those of the highest court in a given jurisdiction are subject to review by higher courts. A party adversely affected by a judicial decision therefore has a strong incentive to discover any error or inconsistency in the document embodying the decision that could be used to attack the decision in a higher court. Moreover, in Anglo-American jurisdictions the doctrine of *stare decisis* permits judicial decisions to be used as authorities in resolving subsequent disputes. The impact of a document may therefore extend far beyond the parties whose dispute gave rise to the document.

The high requirements for correctness and consistency mean that even routine orders in sufficient volume may constitute a significant drain on judicial resources. It is widely recognized that rising case-loads constitute one of the most pervasive problems confronting the judicial system in the United States (Snellenburg 1989). Automation of the drafting of routine judicial documents has the potential to ameliorate the worst effects of these rising case-loads by enabling judges to use their time and expertise as efficiently as possible.

This paper first briefly describes three AI approaches to automated document drafting and sketches the advantages and disadvantages of each. Section 3 then distinguishes among classes of judicial documents based on (1) the stage of the judicial process in which they are created, (2) the complexity of the documents, and (3) the party who drafts the documents: a judge, judicial support personnel, or a litigant. The objective of this section is to identify classes of judicial documents amenable to current AI techniques. Section 4 describes several implemented prototype systems for drafting judicial documents at various stages of the judicial process.

2 Techniques for Automated Document Drafting

Three main approaches to automated document drafting can be distinguished. The first is a *procedural* approach under commands in an imperative programming language select text elements for combination based on user-provided, document-specific data. The earliest document generation systems used this approach, e.g., (Sprowl 1979), and many commercial systems still do so. The advantage of this approach is that it requires only conventional programming tools and techniques.

There are several disadvantages to this approach, however. First, while it is relatively easy to write a computer program that elicits information and generates a character string representing a single document, verification and maintenance of such a program is relatively difficult. *Verification* is the process of insuring that the documents produced by the program satisfy the stylistic and substantive requirements of the document genre. *Maintenance* is alteration of the program to satisfy changes in stylistic and substantive requirements. Verification and maintenance of systems using the procedural approach are difficult because such systems typically lack an explicit, declarative representation of the following:

- The textual components of a document, e.g., paragraphs and clauses,
- The rules for selecting textual elements of the documents, that is, the constraints that exist between textual elements and the facts of each specific document, and
- The goals that the document components are to achieve and the stylistic constraints that they must satisfy.

Instead, each of these elements is implicit in a set of instructions in an imperative programming language.

A second limitation is that procedural systems are typically limited in *scope*, that is, in the range of documents that they can produce. In principle, a procedure could be devised to model any class of documents, but in practice the absence of explicit representation of text elements, constraints between text elements and facts, and document goals makes it impracticable to develop procedures capable of generating heterogeneous classes of documents.

Finally, documents produced by procedural systems lack *informedness*, that is, information about the reasons that textual elements have been included and their function in the document. This makes it impossible to retrieve or compare documents based on their structure and purpose (as opposed to their constituent words) or to determine why a textual element was included in a document created by such a system.

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. Typically, tokens include not just document-specific facts (e.g., <Plaintiff's-attorney-or-attorneys>) but subordinate templates (termed `sub-documents' in CAPS and `submodels' in Scrivenir (Lauritsen 1992)) themselves consisting of text with embedded tokens and tests.

The boundary between procedural and template-based approaches is not crisp: a template with a sufficiently high density of tests can be viewed as essentially a procedure. However, the strength of the template approach is that it makes explicit the structure common to the members of a class of documents. By providing a declarative document model that can be inspected, corrected, and updated, this approach makes system verification and maintenance far easier than under the procedural approach.

However, the template-based approach shares with the procedural approach the weakness of limited scope, because a given template makes explicit only the text and tokens common to all members of a given class. Thus, the template-based approach is declarative only as to classes of documents that share substantial similarity. Moreover, the template-based approach has limited informedness: a trace of the tests that were triggered in creating the document could in principle be retained, but the absence of an explicit model of the purposes that textual elements are to satisfy and the stylistic constraints they must satisfy makes it impossible to retrieve or compare documents based on their structure and purpose or to determine why a textual element was included. Despite these limitations, numerous successful legal applications of the template-based approach have been developed, including automated drafting of pension forecast letters (Spirgel-Sinclair 1988) and management of classes of contracts (Daskalopulu and Sergot 1995).

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*, the study of the illocutionary[1] content of discourse, that is, the goals that speakers seek to accomplish through their discourse (Grice 1975, Searle 1969). The insights of speech-act theory were used in the computational linguistics community to develop techniques for understanding multi-sentential text by inferring the text's underlying discourse structure (Grosz & Sidner 1986, Hobbs 1979, Mann & Thompson 1987). This discourse structure comprises a number of rhetorical relations among sentences, such as elaboration, exemplification, generalization, and sequence.

The success of discourse theory in text understanding systems led to its application to multi-sentential text generation as well (Hovy 1990). Discourse-based approaches to text generation have been used extensively in tutorial systems, e.g., (McKeown 1985, Lester & Porter 1996), for automated generation of software engineering reports, e.g., (Korelsky et al 1993), and for medical patient information reports (DiMarco et al. 1995).

In the law and AI community, the primary focus of the study of discourse structure has been *argument theory*, the study of the relationships within multi-sentential texts that are intended to persuade, justify, establish, or prove. An influential model of argument structure proposed by Toulmin (1958) analyzed argumentative texts in terms of the concepts of warrant, ground, conclusion, backing, and qualification. This model has been used for explanation generation (Zeleznikow & Stranieri 1995) and for document drafting in PLAID (Bench-Capon & Staniford 1995). PLAID produces a document by generating an argument structure whose nodes were tagged according to their role in the argument (claim, rebuttal, support, qualification, etc.). This structure is then pruned to exclude premises that should be implicit in the final presentation and organized into a structure, including linking text, based on a high-level rhetorical template. JEDA (Pethe et al. 1989) and LawClerk (Branting 1993), discussed in more detail below, used a declarative representation of legal rules to organize text templates around a goal tree generated during subgoaling.

Argument theory provides a discourse model for documents whose purpose is to persuade, justify, establish, or prove, but argument theory itself doesn't specify other aspects of the discourse structure of legal documents, such as stylistic and ordering constraints, nor is it applicable to documents with purposes other than to persuade, justify, establish, or prove. A more general approach to discourse structure was proposed in the Docu-Planner (Branting et al. 1997). As described in more detail below, the Docu-Planner uses a discourse grammar capable of generating any document in a given genre. Branting et al. (1997) showed

[1] An illocutionary action is a speech act such as informing, requesting, warning, promising, or pronouncing a man and women to be man and wife.

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.

The discourse-structure approach has the virtue of wide scope, since a set of discourse operators can characterize a wide range of different document structures. The discourse-structure approach also permits relatively easy verification and maintenance, since the discourse operators can be tested and updated individually, and permits a high degree of informedness, since it makes explicit the goals and stylistic constraints that characterize a document genre. However, these advantages come at the price of difficulty in system creation. The discourse approach requires analyzing multiple texts to determine the underlying discourse structure, and then expressing this structure in a discourse grammar. This is a process that requires considerable expertise and investment of time.

The relative advantages and disadvantages of the three approaches to document creation are set forth in Table 1. In general, the template-based systems are easier to create, verify, and maintain than procedural systems without being any narrower in scope or more limited in informedness. As a result, the template-based approach is always to be preferred over the procedural approach. Template-based systems are much easier to create than discourse-based systems, but their scope and informedness is lower. So, the choice between the template- and discourse-based approaches depends on (1) the complexity of the documents, (2) the necessary degree of informedness, and (3) the amount of resources available for system development. Document complexity is an important factor because if the documents are very regular in structure, a wide scope is not necessary, and template-based approaches are sufficient. If there is considerable variation in the documents, by contrast, the wider scope of discourse-based approaches is essential. Similarly, if a high degree of informedness is needed because, e.g., it is important for users to understand why each text element is included or documents need to be compared or retrieved on the basis of their purpose and effect, rather than the particular words that occur in them, then a discourse-based approach is necessary[2]. If informedness in not important, a template-based approach may be sufficient. Finally, a discourse-based approach is feasible only if substantial development resources are available.

Techniques	Creation Difficulty	Verification and Maintenance Difficulty	Scope	Informedness
Procedural	Medium	High	Low	Low
Template-Based	Low	Low	Low	Low
Discourse-Based	High	Low	High	High

Table 1. Comparison of advantages and disadvantages of procedural, template-based, and discourse-based techniques for document creation.

The next section discusses a number of representative judicial documents in terms of their complexity, their role in the judicial process, and the party who drafts them.

3 Representative Classes of Judicial Documents

A judicial proceeding typically consists of a series of steps, each characterized by documents of varying complexity drafted by the litigants, the judge, or judicial staff. The vertical axis in Figure 1 represents the stages of the judicial process, and the horizontal axis represents document complexity. Names of representative documents have been placed in the graph based on their complexity and sequence within the judicial process. Shadowed, shaded, and simple line boxes represent, respectively, documents produced by judges, judicial staff, or litigants.

[2] See (Gordon 1989) and (Lauritsen 1993) for a discussion of the importance of informedness and (Branting & Lester 1996) for a discussion of the role of informedness in maintenance of legacy documents.

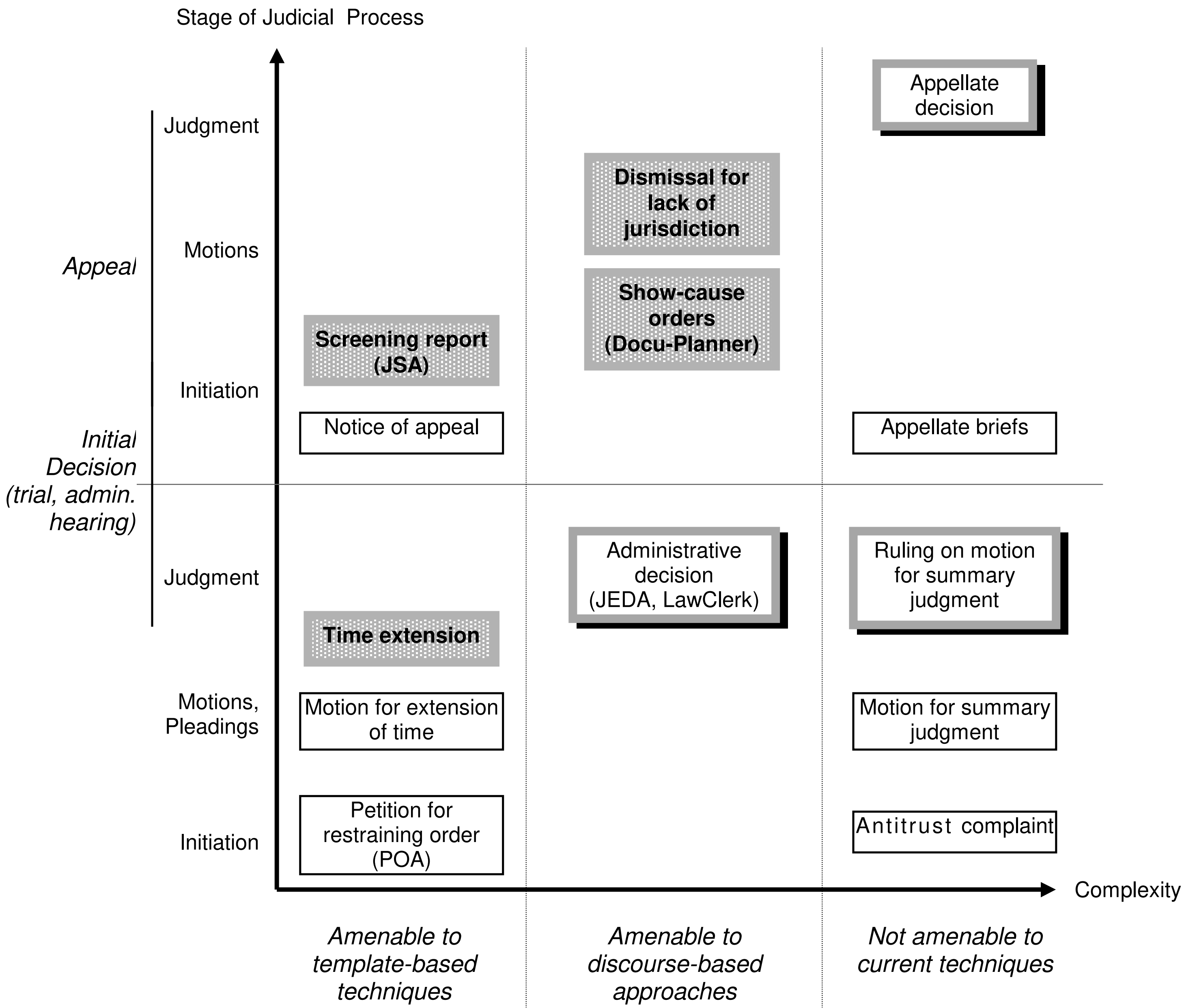
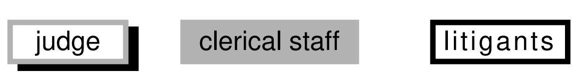


Figure 1. Various representative judicial documents displayed in relation to their complexity and stage in the judicial process. The box surrounding a document indicates whether it is produced by the , , or .



Two stages of a judicial process can be distinguished: the initial decision, e.g., a trial or administrative hearing; and an appeal of the initial decision. The initial-decision and appeal stages each begin with an initiation document, followed by a series of motions or other interlocutory documents and concluded by a judgment.

The documents appearing on the left side of the graph are sufficiently simple that a template-based approach is sufficient for their automation. Template-amenable documents generated by litigants include simple petitions (such as petitions for a restraining order), routine motions (such as motions for extension of time(, and notices (such as notices of appeal). While such documents may sometimes embody a considerable amount of case-specific information, the structures of these documents are very regularly and stereotyped. Template-amenable documents generated by judicial staff include orders granting routine motions, such as motions for extension of time, and jurisdictional screening reports that summarize how an appeal satisfies the requirements for appellate jurisdiction.

In the middle of the graph in Figure 1 are documents that have sufficient structural variation that they are not amenable to simple template-based approaches but require instead some discourse model. For example, administrative decisions have a highly stereotyped, proof-like structure. However, the inclusion and arrangement of particular text segments in the document depends on the particular legal rules that are applicable to the facts of the particular case as decided by the judge or hearing officer. Such documents are therefore much more easily characterized by the argumentation structure that they express than by any text template. Similarly, the structure of documents such as jurisdictional show-cause orders (which order an appeal to be dismissed unless a litigant can rebut an apparent jurisdictional defect) or dismissal orders depends on the legal rules that justify the order. Accordingly, a discourse-based approach that configures text based on the argumentation structure formed from legal rules is an appropriate approach to such documents.

Finally, the right side of the graph shows legal documents that are too complex for current AI techniques. Appellate decisions almost invariably involve facts, rules, or issues that arise only rarely and involve complex discourse structures beyond the ability of current techniques to model. The same complexity and high variability characterizes complaints in some complex domains, such as antitrust law, complex motions, such as motions for summary judgment, and appellate briefs.

4 Examples of Automated Judicial Document Drafting Systems

This section describes several representative systems for drafting documents.

4.1 *Initiation of a Judicial Process by a Pro Se Litigant: Protection Order Petition Drafting*

A promising area of application for automated document drafting techniques is providing assistance to *pro se* litigants. In the U.S., individuals have a right to represent themselves in most legal actions. However, *pro se* litigant typically have great difficulty in drafting documents that conform to the requirements of judicial procedures, and this inability makes such litigants disproportionately burdensome to court personnel.

For example, individuals seeking protection orders or restraining orders against estranged spouses or abusive domestic partners often lack the resources to hire a lawyer to represent them. In U.S. Courts, *pro se* petitioners for protection orders can create great distress for clerical personnel, because these personnel are forbidden to provide legal advice and therefore feel unable to help petitioners who are obviously in distress and may even be in danger.

POA (Protection Order Advisor) is a program developed in cooperation with the Idaho Supreme Court to assist *pro se* petitioners in Idaho courts determine whether they qualify for domestic violence protection orders and, if so, assist them in preparing the petitions. The form of petitions and protection orders is specified by Idaho Supreme Court rules and is quite rigid. This rigid structure is very well suited to a template-based approach.

POA is designed to be run on a computer in a publicly accessible area of a court clerk's office. POA interviews the user in two steps. In the first, POA elicits the facts necessary to determine whether the user is likely to qualify for a protection order. If it appears that the user may qualify, the program elicits information necessary to complete the petition, a sheriff's information report, which is used by the sheriff's department to assist in making service of the petition on the respondent (i.e., the abusive spouse or domestic partner), and the protection order itself. All legal terms appearing in interview windows are hyperlinked to an extensive help file, which can also be accessed through a help menu item. The user can scroll forward or backwards through the interview windows at any time. The petition, sheriff's report, and protection order are printed at the conclusion of the interview and can then be taken to a clerk for filing.

POA's control structure consists of backward chaining through rules for establishing eligibility for a protection order and for acquiring the data necessary to instantiate the form templates. POA's rule engine is implemented in Amzi Prolog, and POA's interface is implemented in Borland Delphi.

To assist systems administrators at each court in tailoring the help information to specific sites and to facilitate updating of the help information in response to statutory or administrative changes, the entire POA help file is contained in an RTF (Rich Text Form) file editable by the systems administrator. The document templates are similarly editable. An evaluation of POA in district courts in three Idaho counties is scheduled for the spring of 1998.

4.2 *Resolution of Cases by Judges: Drafting Routine Administrative Decisions*

POA is designed to assist *pro se* litigants with document drafting necessary to initiate a judicial process. JEDA and LawClerk, by contrast, are systems designed to assist administrative judges or hearing officers in concluding judicial processes by drafting a decision. Administrative agencies are typically characterized by a large volume of similar cases. Each case typically involves a set of predictable issues, although the specific facts and combinations of issues in a given case may vary considerably. A template-based approach is possible for the simplest administrative decisions, but more typical cases exhibit considerable structural variation corresponding to variations in the ways that administrative rules can apply to possible sets of facts. Thus, an argument-based discourse model of administrative decisions is best suited to such cases.

JEDA and LawClerk are argument-based systems for drafting routine administrative decisions. JEDA (Pethe et al. 1989) is a system to assist administrative law judges in drafting decisions on claims for benefits under the federal Black Lung Benefits Act. JEDA uses mixed-initiative data entry: uncontested facts (e.g., the names of the parties, the docket number) and allegations concerning a case (e.g., medical reports, test results) are entered by clerical personnel by filling in a series of entry forms. During a consultation, the system backchains through the applicable legal rules eliciting rulings from the judge on each legal issue that occurs as a subgoal. As each ruling is made, the judge is presented with an editable text box containing an instantiated text template corresponding to the rule. The judge is free to edit or accept this text as is. The final document consists of the concatenation of the instantiated templates with standard boilerplate text, such as the case caption, signature line, etc.

Judge Rippey, an Administrative Law Judge at the U.S. Department of Labor who has used JEDA since 1987, reports that JEDA reduces the time necessary to write decisions on claims for benefits under the federal Black Lung Benefits Act by a factor of at least two. Unfortunately, JEDA has not been accepted by other Administrative Law Judges on Judge Rippey's division (Rippey 1991).

LawClerk (Branting 1993) follows the overall design of JEDA (separation of data entry into uncontested facts that are entered by completing data forms and rulings that are elicited during backchaining through the applicable rules), but uses a declarative representation of rules, issue templates, and entry forms to permit it apply to a variety of different domains and to be modified to accommodate changes in the law or in the idiom of a particular judge or document type.

LawClerk was used to develop the Food Stamp Fraud Consultant (FSFC), a prototype system for Food Stamp Fraud cases developed in conjunction with the Colorado Division of Administrative Hearings. In LawClerk (and therefore in FSFC), when a judge rules on an issue, any templates associated with the judge's truth-value assignment to the predicate are retrieved, instantiated with current case facts, presented to the user for editing, and written to the appropriate section of the written opinion. LawClerk provides standard 'Why' explanations for questions by climbing the goal tree above the current goal.

Unfortunately, the Administrative Law Judges of the Colorado Division of Administrative Hearings decided that adopting FSFC would be disruptive to the Division's procedures, under which decisions are drafted manually with the assistance of secretaries. FSFC was therefore not adopted.

JEDA and LawClerk illustrate that a discourse-based document drafting approach that uses argumentation theory as its discourse model is adequate for routine judicial decisions. However, these systems also illustrate the reluctance of judges to accept automated document drafting tools. In contrast to *pro se* litigants, for whom legal document drafting is a baffling and enigmatic puzzle, judges are likely to have reached their position by having demonstrated considerable facility at legal skills that include document drafting. Inducing judges to adopt automated drafting tools therefore requires that the tools be carefully tailored to judges' habits, needs, and preferences.

4.3 *Document Drafting by Judicial Staff Personnel: Screening Reports and Show-Cause Orders*

Not all judicial documents are drafted by litigants or judges. Case management often gives rise to documents internal to a court, such as screening reports. Moreover, because of rising case loads, many of the more routine orders and notices generated by courts are actually drafted by judicial support staff, such as staff attorneys, even if they require a judge's signature to be effective.

JSA (Jurisdictional Screening Assistant) is a document drafting system currently under development in conjunction with the Colorado Court of Appeals. Jurisdictional screening, the process of determining whether the requirements for an appeal have been satisfied, is an important but time-consuming and relatively tedious function of judicial personnel. Jurisdictional screening is important because it permits cases with jurisdictional defects to be recognized as soon as possible, minimizing the judicial resources that are consumed by such cases.

The Colorado Court of Appeals receives an average of approximately 100 new cases per month. Screening these appeals is too complex for clerical personnel, but must instead be performed by a staff attorney. A staff attorney fully trained in screening spends approximately 50% of his or her time screening, but much of the screening is done by attorneys who are not fully trained. This is because few staff attorneys are willing to do jurisdictional screening for more than six months, and many do screening for as little as three months. As a result, several attorneys must learn jurisdictional screening each year, a process that takes approximately two months. An attorney who has worked less than two months in screening may spend up to 100% of his or her time on this task, leaving little time for other activities.

JSA is intended to automate two aspects of jurisdictional screening: determining the jurisdictional rules and precedents applicable under the facts of a particular case and the consequences that follow from those rules and precedents; and generating screening reports and routine show-cause orders.

JSA uses a different human-computer interaction model than JEDA or LawClerk. JSA presents the user with an electronic screening-report form. When the user moves the cursor to a field, the required datatype of the field is displayed in a status bar. Menu items for each field include examples, counter-examples, "how-to" advise, and "consult." Selecting the "consult" option invokes the rule engine, which then backchains through the applicable rules, eliciting additional information through pop-up windows if necessary. By permitting the user the option of either providing field values him/herself or invoking the inference engine for assistance, the system places the level of automated assistance under the control of the user. This permits a novice user to be interviewed by the system, but enables an experienced user to simply enter information that the user can determine him/herself into the appropriate fields without the delay of being interviewed.

When the user has entered the facts necessary to determine whether the requirements for jurisdiction are satisfied, JSA drafts a screening report that summarizes this information. If there is a jurisdictional defect, JSA is also designed to draft simple show-cause orders ordering the appellant to remedy the defect or suffer dismissal.

JSA uses a template-based document model. This is appropriate for screening reports because screening reports are simply catalogs of facts and legal conclusions relevant to jurisdiction. The simplest show-cause orders are also amenable to a template-based approach. However, because of the wide variety of different factors that can lead to jurisdictional defects, complete coverage of show-cause orders requires a discourse-based approach.

The limitations of a template-based document model for show-cause orders led to the development of Docu-Planner (Branting & Lester 1996, Branting et al. 1997), a system that uses a full discourse model of document genres, including both illocutionary and rhetorical operators. The set of these operators constitutes a *document grammar* capable either of parsing existing documents in the genre or of generating new documents. A prototype grammar for appellate jurisdictional show-cause orders was implemented in Docu-Planner. This grammar was based on the observation that a performative judicial document, such as an order or decision, must satisfy three requirements if it to define or alter a legal relations. First, the document must *find* that some set of relevant facts is present in the case. Second, the document must *rule* that one or more legal propositions follow from applicable legal authorities under the facts. Finally, the document must *order* some legal consequence justified by the legal propositions under the given facts. The document grammar for show-cause orders includes these illocutionary operators together with a set of rhetorical operators that express the stylistic constraints of show-cause orders.

Docu-Planner uses a unification-based formalism to synthesize new documents from a discourse grammar. As described in (Branting & Lester 1996) and (Branting et al. 1997), the document grammar for show-cause orders achieves a high degree of scope and informedness. However, a methodology for developing document grammars is still a research issue, and tools to help automate this process have yet to be developed. Until this methodology is better understood and until automated document grammar development tools become available, the document grammar approach will be confined to the laboratory.

5 *Conclusion*

This paper has described three AI approaches to automated document drafting, distinguished several classes of judicial documents based on their order of appearance in the sequence of events in the judicial process, their complexity, and their expected users, and described several prototype judicial documents drafting systems. This paper has argued that template-based systems are appropriate for judicial documents with a high degree of homogeneity, such as protection-order petitions and jurisdictional screening reports. Discourse-base approaches are needed for more complex documents. The feasibility of constructing discourse-based systems that use argumentation theory as their discourse model has been demonstrated in several systems, including JEDA and LawClerk. With several implemented prototypes, systems of this type are increasingly well understood.

Docu-Planner illustrates that more powerful document grammars integrating a variety of illocutionary and rhetorical operators can be developed based upon current discourse theory. However, further research is needed to develop knowledge acquisition tools for document grammar creation. The lack of acceptance of automated decision drafting systems appears to be the result less of the limitations of the computational models underlying these system than of the pragmatic difficulty of inducing judges to use automated document drafting tools. By contrast, *pro se* litigants appear to be very receptive to automated assistance. Moreover, judicial support personnel appear more receptive to automated tools than judges because the documents they draft tend to be more routine and because they tend to be less experienced at drafting such documents than judges.

As the utility of automating the more routine aspects of document drafting within the judiciary becomes more widely recognized, I believe that automated document drafting will become a routine component of the judiciary's technological tool-kit. The systems described in this paper represent initial steps towards an Artificial Intelligence infrastructure that will make the judiciary more accessible and efficient without compromising judges’ full exercise of their judgment and expertise.

Acknowledgments

This research was support in part by grants the National Center for Automated Information Research, the Idaho Supreme Court, and by NSF Faculty Early Career Development Grant IRI-9502152. Patrick Broos assisted in the development of LawClerk, Hollis Mariott assisted in the development of JSA, and Susan Hanna performed the programming of POA. Docu-Planner was developed in collaboration with James Lester and Charles Callaway.

References

T. Bench-Capon and G. Staniford, `PLAID - Proactive Legal Assistance' (1995*) Proceedings of the Fifth International Conference on Artificial Intelligence and Law (ICAIL-95),* College Park, MD, ACM Press, pp. 81-88.

L. K. Branting, 'An Issue-Oriented Approach to Judicial Document Assembly' (1993) *Proceedings of the Fourth International Conference on Artificial Intelligence and Law (ICAIL-93),* Amsterdam, The Netherlands, ACM Press, pp. 228-235.

L. K. Branting and J. C. Lester, `Justification Structures for Document Reuse' (1996*) Proceedings of the Third European Workshop on Case-Based Reasoning (EWCR-96)*, Lausanne, Switzerland, pages 76-90.

L. K. Branting, J.C. Lester, and C. Callaway. `Automating Drafting of Self-Explaining Documents' (1997) *Proceedings of the Sixth International Conference on Artificial Intelligence and Law (ICAIL-97),* Melbourne, AU, ACM Press, pp.72-81.

A. Daskalopulu and M. Sergot, `A Constraint-Driven System for Contract Assembly' (1995) Proceedings of the Fifth International Conference on Artificial Intelligence and Law, ACM Press, pp. 62-70.

C. DiMarco, G. Hirst, and L. Wanner, `Health Doc: Customizing Patient Information and Health Education by Medical Condition and Personal Characteristics' (1995*) Working Notes of the Workshop on Artificial Intelligence in Patient Education*.

T. Gordon, `A Theory Construction Approach to Legal Document Assembly' (1989*) Pre-Proceedings of the Third International Conference on Logic, Informatics, and Law* , Florence, Italy, pp. 485-498.

H. Grice, `Logic and Conversation' In P. Cole and J. Morgan eds., *Syntax and Semantics 2: Speech Acts*, pp. 41-58. (Academic Press: New York, N.Y. 1975).

B. Grosz and C. Sidner, `Attention, Intention, and the Structure of Discourse' (1986) *Computational Linguistics* 12(3).

J. Hobbs, `Coherence and Co-reference' (1979) *Cognitive Science* 3(10):67-82.

E. Hovy `Pragmatics and Natural Language Generation' (1990) *Artificial Intelligence* 43:153-197.

R. Kittredge, T. Korelsky, and O. Rambow, `On the Need for Domain Communication Knowledge' (1991) *Computational Intelligence*, 7(4):305-314.

T. Korelsky, D. McCullough, and O. Rambow, `Knowledge requirements for the automatic generation of project management reports' (1993) *Proceedings of the Eighth Knowledge-Engineering Conference*, IEEE Computer Society Press.

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

M. Lauritsen, `Knowing documents' (1993*) Fourth International Conference on Artificial Intelligence and Law (ICAIL-93*), Amsterdam, The Netherlands, pages 185-191, 1993. ACM Press.

J. Lester and B. Porter, `Scaling Up Explanation Generation: Large-Scale Knowledge-Based and Empirical Studies' (1996*) Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)* (AAAI Press) pp. 416-423.

W. Mann and S. Thompson, `Rhetorical Structure Theory: A Theory of Text Organization' (1987) Technical Report ISI/RS-87-190, USC/Information Sciences Institute, Marina del Rey, CA.

K. McKeown, `Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text' (Cambridge University Press: Cambridge 1985).

V. P. Pethe, C. P. Rippey, and L. V. Kale, `A Specialized Expert System for Judicial Decision Support' (1989) *Proceedings of the Second International Conference on Artificial Intelligence and Law (ICAIL-89)* Vancouver, B.C., pages 190-194.

Judge Charles P. Rippey, Personal communication (1991).

J. Searle, *Speech Acts: An Essay in the Philosophy of Language* (Cambridge University Press: Cambridge 1969).

A. Snellenburg, 'New Approaches to Reducing Court Delay and Congestion' (1989) *State Court Journal* 13(3).

S. Spirgel-Sinclair, `The DHSS retirement pension forecast and advice system' In (P. Duffin, ed.) *KBSs in Government* 88, pp. 89-106. Blenheim On Line, Pinner, 1988.

S. Toulmin, `The Uses of Argument' (Cambridge University Press: Cambridge 1958).

J. Zeleznikow and A. Stranieri, `The Split-Up System: Integrating Neural Networks and Rule-Based Reasoning in the Legal Domain' (1995) *Proceedings of the Fifth International Conference on Artificial Intelligence and Law* (ACM Press) pp. 185-194.