Toward a General Theory of Document Modeling

Marc Lauritsen
Capstone Practice Systems
Harvard, Massachusetts
marc@capstonepractice.com

Thomas F. Gordon
Fraunhofer FOKUS
Berlin, Germany
thomas.gordon@fokus.fraunhofer.de

ABSTRACT
Most legal tasks involve document preparation and review. Drafting effective texts is central to lawyering, judging, legislating, and regulating. How best to support that work with intelligent tools is an ancient topic in AI-and-Law research. For those tools to work, they must have good quality knowledge content to work with. Many alternative theories and techniques for modeling documents have been developed for particular kinds of situations. This article sketches a basic general theory of legal document modeling, with a focus on the key role of argumentation.

1. INTRODUCTION
Legal work is a dance of knowledge, deliberation, and action. Among the most common kinds of action in that dance are text preparation and review. Documents play into most of what lawyers know and do. One important form of legal knowledge representation is accordingly the modeling of documents.

What are document models? Those discussed here are best understood as collections of statements about what should and shouldn't be the case with respect to a document of a certain type. To model a document is to set forth some of the characteristics it should have. Models accordingly always express normative positions.

This article is organized as follows: Section 2 describes the contexts within which most contemporary legal document preparation occurs, namely drafting "systems" in which people interact with texts, machines, and each other. Section 3 covers some of the basic concepts and phenomena encountered in such contexts. Section 4 introduces our theory of document models, following a review of related work. Section 5 explores how models of the sort we describe can be used in practical drafting processes. In section 6 we make observations about our theory and consider to what extent legal document modeling is distinctively legal. Section 7 concludes the article and describes potential future work.

The basic phenomena of document modeling are reasonably straightforward, but disclose considerable complexity when closely examined. To our knowledge, no one has yet thoroughly described the characteristic structures and operations involved. We believe the material here goes beyond what has previously been written about legal document modeling, introduces a valuable new perspective that recognizes the central role of argumentation in such modeling, and provides a solid foundation for both theoretical and practical progress.

2. THE CONTEXT: DRAFTING SYSTEMS
Models are intended to play a role in the process of preparing and reviewing documents. For legal documents, lawyers generally refer to that process as drafting.

A drafting system consists of

- one or more people engaged in the composition and review of texts,
- one or more collections of texts, some of which model how texts should be composed,
- software processes operating on those collections (autonomously or at the direct behest of people), and
- physical devices that render, store, process, and interact with the texts.

2.1 Functions
Drafting typically involves working *on* texts *with* texts.

A legal drafting system helps people create texts by (1) enabling them to search for, browse, read, and copy existing materials and guidance, (2) accepting new texts, some of which say things about existing texts, and (3) generating texts of various sorts, some of which may change as users interact with them.

2.2 Knowledge components
A wide range of knowledge components are encountered in the forms and commentaries that one finds in conventional drafting systems. For example:

- What issues need to be dealt with, and considerations on their potential resolutions
- Who knows about the subject matter
- What words best go where when
- Variations to consider (aggressive stances, compromises)
- What to expect from other parties

- How to react to a counterparty's rejection of certain provisions (what 'fallbacks' to consider offering)
- How to gather needed information and raw material
- Questions to get answers to (from the client or elsewhere)
- Stylistic rules and conventions

Some forms of knowledge aren't well communicated in flat, static documents. Explicitly modeled content – with variability, conditionality, repetition, and annotation – serves much better, particularly when leveraged in automated processes.

2.3 Varieties of commentary

Many forms of commentary are involved. They may be about a kind of transaction and its characteristic documents in general, or specific to a given matter.

Commentaries come in different *kinds*, such as:

- explanation (of history, purpose)
- guidance (suggestion, recommendation)
- notes about alternative wordings
- links to auxiliary resources

and have different *subjects*, such as

- a transaction as a whole
- a particular document as a whole
- a specific issue or decision (transactional or drafting)
- a particular passage or context

They can be *accessed* in different places and ways. Some naturally are associated with locations in a form, some with questions to be answered, some with both. Commentary text may need to vary based on whether it is in context or in a standalone compilation. It may be *stored* in the system or in external sources such as websites or databases. Resources may include links to other memoranda in an organization's file system, intranet or web pages, or to experts who can be contacted.

2.4 Tasks

There are also many different kinds of things to do, consider, and decide in the work of drafting legal documents. Some are unconditionally suggested in any transaction of the sort covered, others are conditionally suggested based on transactional characteristics. Some are intra-documentary (specific to a particular document); some are inter-documentary (relating to more than one); others are extra-documentary (pertaining to issues and actions outside of any document).

2.5 Triggers for variation

There are many different triggers for variation in a drafting system, any of which may impact model language, associated commentary, or suggested issues/actions. For example, such triggers in a law practice setting may include:

- client type and identity
- counterparty type and identity
- counterparty counsel identity
- industry
- transactional terms
- transactional *events* (such as a position taken by a counterparty during negotiation)
- user preferences

2.6 The actors and actions

Two kinds of actors interact with the above system – people and software agents. It's useful to remind ourselves what kinds of things each does.

2.6.1 People

Document-related actions most associated with human actors include:

- finding examples and other raw material (both in-system and in the outside world), such as
  - similar transactions
  - similar whole documents and sets
  - similar components, such as clauses
  - people with relevant experience
- stitching pieces into a new draft
- replacing old transaction-specific material with analogous material for a new transaction
- composing fresh texts
- negotiating (proposing, reacting, arguing)
- revising
- comparing
- modeling
- proofreading, checking, or otherwise reviewing
- critiquing or otherwise commenting on a draft

Note the knowledge tasks going on. People need to know and decide *what* to say, and *how* and *where* to say it. And they need to know what one needs to know and decide in order to do the above things.

2.6.2 Software

Software can perform a wide range of tasks in a drafting system:

- present texts and text-like interfaces in different renderings
- elicit and accept input
- inform and educate
- supply links to useful resources
- format
- assemble
- record
- learn
- notice gaps, conflicts, and ambiguities in documents and models
- interpret and translate

Software can serve as an observer and text maker/editor – a speech-actor in its own right. It can act as a model enforcer, manager, or intelligent assistant. When operating contemporaneously with a human editor, software can exercise different degrees of initiative:

- watching and recording without intervening
- alerting and reminding
- suggesting changes and volunteering possibilities
- preventing certain changes
- making certain changes (e.g., to maintain model compliance or semantic consistency)

2.7 Purposes and opinions

Documents are usually intended by their authors to serve one or more purposes. Particular parts or features of documents are likewise intended to serve one or more purposes. People often make differing claims and hold differing beliefs about what purposes a document (or a feature or part thereof) is likely to serve, or should seek to serve.

Most legal documents also are *performative*, in the sense that they express speech acts such as promising, agreeing, informing, claiming, finding, and ruling. Such "illocutionary" goals are typically among the primary purposes a document is designed to achieve.

Documents can belong to one or more classes of documents, understood as groups that share features. People often make differing claims and hold differing beliefs about what features a document (or a part) exhibits and therefore what class(es) it belongs to. When formal methods are used to construct documents, such as XML document schemes, some of the classes a document belongs to are clearly determinable. It belongs *at least* to the class of instances of its schema. One could however continue to disagree about whether it belongs to other classes that have not been defined using formal methods, such as the class of all "persuasive" documents.

Statements about documents are almost always made in natural language, and often have missing or ambiguous attributes or components. The applicability or very meaning of a modeling statement can be the subject of differing claims and beliefs.

2.8 Texts and metatexts

Drafting systems are usefully conceived as involving richly interwoven fabrics of texts and metatexts (texts about texts). This textbase is an implicit network of objects, a dynamic collection whose nodes and links change as users interact with it.

Three basic kinds of texts are involved:

- texts intended for some purpose outside the drafting system (including both finalized documents and those presently in progress) (the objects of regular work);
- textual models and commentaries of various sorts, which are *about* other texts; and
- texts that otherwise support drafting tasks, such as scripts, plans, plan models, and interface definitions.

Our focus in this article is on a subset of the second type above.

3. CONCEPTUAL PRELIMINARIES

3.1 Basic distinctions

Looseness of terminology (such as 'logic' and 'field') and conflation of differences (such as those among different hierarchies and non-hierarchies – composition, containment, condition nesting, kind-of, version-of) can cause considerable confusion in conversations about document modeling. Some of that confusion can be avoided by recognizing such fundamental distinctions as:

- In and About – what is said in (or by) a text vs. what is said about a text.
- Is and Ought – what is (or isn't) in a text vs. what should (or shouldn't) be in it.
- What and Where – what is or should be included (somewhere) vs. where it does or should appear.
- Document models and process models – what a document should say or look like vs. how you should go about the process of drafting or negotiating it.

Texts can be marked up with special tags to signify locations or passages about which something is being said, but such tags are not really "in" the text itself from a logical point of view. They signify the intended object of some *other* text, a metatext, while collapsing both into a single virtual layer.

Current word processing and document assembly tools make it difficult to express those differences. Word processors, for instance, typically collapse text and metatext (e.g., comments) into a single artifact. HotDocs[1] and other assembly engines don't let you easily express that something should occur *somewhere* in a document, without a relative location being specified. Such engines don't supply facilities through which locations, sequences, or containment relationships can be non-deterministic but nonetheless constrained.

3.2 Documents and texts

3.2.1 Document

A document is a bounded collection of one or more texts, recognized as a discrete object by some person for some purpose. It is a kind of record or container used to store, identify, or locate texts. It can exist as a physical object, an electronic file, or an object in memory. For purposes of this article, we take an extremely broad view of 'documents,' and include textual objects like email messages and spoken passages.

3.2.2 Text

A **text** is a consecutive set of characters or other graphemes (visual units of potential linguistic significance.)[2]

---

[1] HotDocs (http://www.hotdocs.com) is the most widely used legal document assembly engine at present. Others include D3 (www.microsystems.com/d3), DealBuilder (www.business-integrity.com), and Exari (www.exari.com). See [18] for a fuller list, and [15] for a historical look at this genre of software.

[2] For present purposes we will mostly talk in terms of texts that can faithfully be captured in linear (one-dimensional) sequences. Of course many documents leverage two-dimensional layouts to express meaning, and such higher-dimensional orderings need to be taken into account in a full theory. We also leave aside images and other non-linguistic elements that might be present in a document.

Most texts consist of many sub-texts, lesser regions of contiguous characters. For instance, the text 'ABCDE' includes the following fifteen texts, any of which might be the subject of attention:

A B C D E

AB BC CD DE

ABC BCD CDE

ABCD BCDE

ABCDE

Texts and subtexts thus form a network of containment relationships.

These can also be understood as falling into this natural hierarchy, expressed as an outline (omitting duplicate nodes):

- ABCDE
  - ABCD
    - ABC
      - AB
        - A
        - B
      - BC
        - C
    - BCD
      - CD
        - D
  - BCDE
    - CDE
      - DE
        - E

Since the number of contiguous sub-sets of a text of N characters is $(N^2 + N)/2$, all non-trivial texts have quite a few. For instance, a text of 100 characters has 5,050. A text of 10,000 words averaging five letters, together with an equal number of spaces and punctuation, has $(60{,}000^2 + 60{,}000)/2$ or 1,800,030,000 sub-texts. For each character added to a text of length N, N+1 new subtexts become specifiable.

The vast majority of sub-texts are of no interest whatsoever. But optimal text editing and modeling require that any arbitrary sub-set be addressable, down to the individual character or glyph. The inability of most commercial legal document management systems to address texts below the document level, or some document assembly systems to address objects below the paragraph level, has posed a major limit on expressiveness.

3.2.3 Metatext

A **metatext** is any text that makes a statement about another text, within or across documents. A text can also say something about itself, or about another metatext. It can be about a specific text, or a defined class of texts.

The things that can be said about text fall into many categories:

- topical or semantic (what the referred-to text is about)
- source (where it came from)
- appearance or formatting (how text segments are formatted or laid out, e.g., in bold, or in a footnote)
- structural (what part of a part-of hierarchy a segment plays, e.g., article, section)
- purposive (the legal or business requirements a text is intended or appears to serve)
- explanatory (for instance why a passage is phrased a particular way, or its function in a document)
- model compliance (to what extent a text satisfies a specified model)

Some kinds of metatexts are peculiarly at home in intelligent drafting systems. These include:

- logical designations (variable fields; conditional or repeating passages)
- text models (descriptions of required and permitted elements, e.g., using XML DTD or schema syntax)

3.2.4 Versions

A version of a text comes into existence when it is copied into a fresh object that can independently be edited. You can also imagine each change in a document under edit to result in a separate virtual version. But only texts that are affirmatively saved as discrete objects constitute versions in common understanding.

Different *renditions* of a document likewise can be regarded as different versions. An italicized character is a different character than a non-italicized one. Even font differences can have intended significance. Each state of a text that involves any difference in its contents is a potentially distinct version. When someone adds, changes, or deletes a character, every supertext containing the location at which that addition or deletion occurs is changed.

All texts, including metatexts, can be versioned. Deciding which metatexts of a given text can appropriately be applied to a new version of that text raises both accounting complexities (computing the correct addresses of the referent texts in the revised object) and semantic challenges (when does a tiny text edit render the prior commentary invalid?)

3.3 Document descriptions and models

It is useful to distinguish two main kinds of documents about documents.

A document *description* is a set of statements about an existing document and its component texts.

A document *model* is a set of normative positions (modeling statements) about what should or should not be true about documents of a particular kind. It makes claims about what can be said about documents that satisfy it. (For example, "This document follows the official recommendations for non-disclosure agreements to be executed by ABC Corporation.")

Descriptive statements have to do with what *is* the case for an existing document; modeling statements have to do with what *should* be the case about a document that may or may not yet exist. Document models take positions on what descriptions ought to be true about documents of a particular class.

Documents are naturally self-documenting in a trivial sense – they are what they are, with letters, words, paragraphs, and

punctuation in the places they appear on their faces. But they almost always communicate meanings and serve purposes beyond their raw content. A paragraph, for instance, is generally something more than just a region of text separated from another by white space or a tab. Captions and formatting conventions are generally used to express structural and functional roles of textual components. Documents presuppose shared understandings about what words and their configurations mean.

A document can never be exhaustively described because no finite being can know all the purposes it might serve, roles it might play, or relationships it might have with other texts.

Models can be used for purposes both of generating and assessing (or auditing) documents. They function as texts that guide the composition and review of other texts.

3.4 Representations and formalisms

A great variety of techniques exist for communicating document models.

In the law practice world, document models are often expressed via model forms with blanks, designated alternatives, and other kinds of markup.

Sometimes an example is used as a model – "documents of this kind should be like *this*" – with the salient similarities left as an exercise for the drafter. Lawyers often draft from the most similar prior transaction, and rely on their intuition to determine what to preserve and what to change.

Courts and regulatory bodies enunciate models by promulgating forms and associated instructions.

Word processing and document assembly programs offer a variety of features to model documents, such as fields and styles. A "template" author can draw on those features to express how a document of a certain type should be composed for purposes of specific transactions, and implement automation to facilitate such composition.

Markup schemes like SGML and XML are quite useful insofar as they are both machine and humanly readable. [22] They provide a means to enrich a document with meta-data, which can be used by software to help authors write documents that conform to particular types, schemas, or, as we call them here, "models". A further advantage is that they separate the content, meta-data and layout (formatting) layers of a document, allowing these to be processed separately. For example, content can be easily translated into multiple presentation formats, or formatted using the style guidelines of a publisher, without having to modify the source document.

Some modeling statements can be expressed in XML DTDs and schemas. Lauritsen [17] conducted an early exploration of how documents could be modeled using XML syntax. Gordon [12] modeled boilerplate using SGML. XML content models fall short for general purpose document modeling, however, in ways that will become clear below. A less formal "concept map" approach to modeling document-related practice knowledge is outlined in [18].

The Web ontology language OWL[3] offers features that serve as a checklist of what will likely be required for an adequately expressive document modeling formalism, such as existential and universal quantification, cardinality restrictions, class disjunction, subsumption relationships, sub-properties, and property chains.

A formalism that involves separating texts and metatexts into an explicit network of aboutness relationships was developed by one of us at a 2007 ICAIL workshop [19]. The same paper reviews the many 'frontiers' that remain in the legal document automation world. We believe that progress on most of them requires the kind of foundational work described in this paper.

4. A MODEL OF DOCUMENT MODELS

4.1 Related work

The common approach to automating legal documents is still procedural. James Sprowl developed one of the first systems of this kind, the ABF Processor [24]. Essentially, the ABF Processor provided a way to embed procedural code for selecting and instantiating "boilerplate" texts (templates) within a model document. The model document became a program which, when executed by the ABF Processor, engaged the user in a dialog to supply values for variables. The processor used these values to select and instantiate the templates. The output of the processor was a formatted document that could be edited and printed using a word processor. Although such applications could in principle be implemented using any conventional programming language, Sprowl's achievement was to develop a domain-specific language for documents that made it feasible for lawyers, the persons with the necessary legal expertise, to develop such applications, without the assistance of computer programmers.

A shortcoming of Sprowl's procedural approach is that it fails to cleanly separate models of substantive law from knowledge about how to produce a document for a specific legal task. This make it difficult to validate the model, maintain it as the law changes, or explain the choices made when constructing a document. The KOKON system [15] was a document assembly system that separated these two components of the application and provided a way to model the legal domain using logic programming techniques, as pioneered by Sergot, et. al. [23].

KOKON is founded on a deductive conception of legal reasoning, where legal rules are applied to facts to derive legal consequences. While there are useful application scenarios of this model, it is based on strong assumptions that in general do not hold. In general legal reasoning is an argumentation and theory-construction process, where deduction plays an important but subordinate role. Gordon [12] was the first to design a legal document assembly system based on a theory-construction conception of legal reasoning, preserving the benefits of KOKON's separation of domain and document models but on the basis of a more comprehensive account of legal reasoning.

Bench-Capon and Staniford developed a document-assembly system, PLAID [1], that used Toulmin's argument scheme [25] to assemble documents for explaining the conclusions of a legal expert system. The legal domain was modeled as a logic program in which the conditions of the rules were annotated by their role in Toulmin's scheme.

At around the same time, Daskalopulu and Sergot [8] developed a system for assembling contracts that used constraint-satisfaction methods to help drafters to select boilerplate texts and arrange them in a document outline in a way which satisfies

[3] www.w3.org/2007/OWL

a given set of constraints. The system did not model knowledge of the substantive law separately from knowledge about how to produce a particular class of legal document for some specific task.

Branting and his colleagues developed a document-assembly system, DocuPlanner [4,5,6,7], using methods from computational linguistics to model the illocutionary and rhetorical structures of legal documents. The illocutionary structure models dependencies between components of a document and the purpose of the document, given the goals of the author. This illocutionary structure models the legal domain, speech acts of the relevant dialogue type from the application scenario, as well argumentation structures expressing dependencies between issues, propositions and arguments. The rhetorical structure models the style and form of the document, given the conventions of the relevant legal community. Together, the two constitute a document grammar that can be used both to generate and explain documents.

Moens [21] provides a useful review of the history of drafting systems, with special attention to those designed to support the production of legal "sources" in the sense of legislation, regulations, doctrine, and court decisions. In such contexts, the retrieve-ability and machine process-ability of system outputs take on special importance.

Hafner and Lauritsen [13] developed the idea of document models as trees of text, sub-trees, and choice nodes, and explored how such models could be unified when multiple ones needed to be followed.

Various initiatives are underway, such as "Norme in Rete" in Italy [20] and the MetaLex working group of the European CEN standards body [3], to develop XML document schemes for legislation, along with tools for helping to write, annotate, store, browse and search legislation in these formats. See [2,9,10,11, and 14].

4.2 Document models

A **document model** is a set of modeling statements, along with one or more claims about documents that satisfy those statements. It may also include statements about itself and/or about the modeling statements it contains. You can think of it as a "theory" in the sense of a set of propositions.

A model as a whole describes and defines a single class of documents (those that comport with it.) As such, it can be regarded as a class definition. Documents that satisfy it can fairly be said to be in the class it defines.

The null model contains no modeling statements, and therefore is satisfied by all documents.

Model attributes include completeness and consistency. Few models are complete in the sense of fully specifying the attributes of documents. And especially when models are composed from disparate sources, inconsistencies can arise, in the sense that modeling statements may call for attributes of a document that cannot simultaneously be achieved.

Models do not typically model perfection, just some degree of goodness, which is relative to the modeler's goals and the drafting circumstances, and is a matter of argument.

While they may be put to procedural uses, document models are declarative structures.

4.3 Modeling statements

A **modeling statement** is a normative proposition about one or more characteristics that documents of certain kinds should or shouldn't have. It is a normative position taken by one or more natural persons as of a particular time.

Each modeling statement has an (implicit or explicit) *scope of reference* – the kind(s) of documents it purports to speak about.

Statements can have *sources* or *authors* (and potentially *endorsers*) and *as-of dates* or ranges. They are implicitly addressed to authors of documents intended to be in the class of documents described by the statement

Each has a *deontic intensity* – the extent to which the described characteristic should or should not be the case. This can be expressed for instance as points in a spectrum of advisability like the following:

Required

Strongly advised

Advised

Mildly advised

Permitted

Mildly unadvised

Unadvised

Strongly unadvised

Prohibited

Note that modeling statements expressed in the imperative voice ("Include this sentence here under these circumstances") may leave the intended deontic intensity ambiguous.

Modeling statements can speak to different *aspects* of documents, such as

- Which texts should be included somewhere (occurrence)
- Which sequence or structural position texts should follow (location)
- What wording can reasonably be expected to accomplish a given legal or strategic result

Anything that can be said about a document (or part or aspect) can be the subject of a modeling statement, but usually such statements are limited to document features that are at least potentially within the control of the drafter.

The expressed characteristic may be *conditional* on circumstances. For example, a conflict of laws provision may be strongly advised if the parties are based in different jurisdictions.

Statements can have different degrees of specificity, precision, and formality. Whether a given document satisfies them accordingly can be a matter of opinion.

Modeling statements may be indirect. A statement can for instance assert that a document should include text that reasonably can be expected to accomplish a particular purpose, without specifying that text. Or that it should be able to be defendably described as having a certain attribute (such as readability.) Thus some of the very features modeled relate to imagined argumentative survival.

Modeling statements may also have an associated 'backing' – *why* a particular characteristic is advised to a certain degree for documents of this kind. Such reasons are typically in reference to one or more legal or strategic purposes that documents of the modeled class are intended to serve.

Document models involve *statements about statements*, often having to do with opinions about which texts in a document or a part of it are reasonably likely to accomplish an intended legal or strategic purpose. They also typically involve mixtures of illocutionary, deontic, probabilistic, teleological, and pragmatic expression modes that are challenging to unpack. A theory of document modeling should accommodate even poorly formed, vague, and self-contradictory statements.

4.4 Examples

Here are paraphrases of some document modeling statements, to illustrate the range of forms:

- An ICAIL paper should not exceed 5000 words.
- A nondisclosure agreement should include a clause about its duration.
- All of our contracts should refer to us as 'ABC Corporation, a Delaware corporation.'
- Defined terms must be capitalized.
- Include Section XXI only if the counterparty is Stanford Law School
- Consider adding appendices for intellectual property explicitly conveyed and not conveyed in the license agreement.
- Make sure you don't use the phrase "open source" in any of our software agreements.
- An order to show cause why an appeal should not be dismissed should contain statements establishing that the notice of appeal was filed after the date it was due.
- License agreements intended to be accepted by consumers should have a Flesch-Kincaid readability score of 12 or less.
- There should be separate appendices for each of the guarantors.
- Be sure that the letter of understanding makes clear what the parties' respective rights are in the event an act of war makes performance impossible.
- The noncompetition clause should withstand scrutiny under the standards articulated by the Massachusetts courts.
- The first sentence of an employment agreement with ABC Corporation may consist of the following text, with placeholders replaced by the relevant specifics: "This Employment Agreement, by and between ABC Corporation and «Employee Name», is entered into on «Agreement Date»."
- Use "will" not "shall" to communicate the future tense.

There are also innumerable tacit directives to which drafters need to be responsive, such as 'don't misspell words' and 'use proper English.'

The examples above mostly relate to drafting of the sort that occurs in a law office in connection with business transactions, but the principles are generalizable to other contexts. Our goal is to formulate a conceptual framework that works for *any* kind of legal text composition.

5. MODELS IN USE

5.1 Using a model in a drafting process

By keeping track of which models a document in draft is seeking to satisfy, and what aspects of those models can be said to be satisfied or not, a drafter can effectively ascertain what work remains to be done, and what aspects of the draft may require revisiting.

These considerations imply that three objects will generally be in play:

1. The model intended to be followed;
2. The document in draft; and
3. A set of statements by one or more persons about which components of the model are presently satisfied.

Models whose prescriptions are conditional on the circumstances of a particular transaction can be instantiated to reflect just those that apply to that transaction, thus sparing the drafter unnecessary material to pay attention to.

Real drafting almost always involves integrating prescriptions from a variety of sources. Such virtual or composite models may also change dynamically as the sources and circumstances change.

Document generation may be regarded as a theory and argument construction process in which the model (the "theory") may need to be changed during the process of generating a specific document, in the light of new information, made available by the specifics of the case, which sheds new light on the document model and its rationale. Just as courts reinterpret legislation in the light of the facts of specific cases.

5.2 Formal models and practical modeling

While formalizing models in the terms described above may provide maximal expressiveness and context-independence, practical document modeling work may proceed more easily with alternative representations. For instance, a passage of text marked up with variable placeholders and delimiters to signify conditional and repeated sub-passages will often be a more economical way – a kind of shorthand – to express an intended model than a list of explicit modeling propositions.

We intend to explore technical solutions that preserve the practicality and ease-of-use of conventional document assembly systems while at the same time enabling the services possible with our richer conception of document models.

5.3 Practical differences

Modeling and drafting legal texts within systems based on models as conceived here would be substantially different from today's practices. Here are some of the differences, and aspects of how we imagine this working.

- An architecture that separates texts about texts from the text they are about enables *anyone to say anything about anything at any time*, without concerns about file locking

and collisions. Commentary and modeling become fully distributed activities. They can occur as episodically, incrementally, incompletely, and informally as people like. Many voices can be heard, and they need not all agree. Drafters can use as much or as little of that as they wish. Yet both people and software agents can draw such materials into operative models when appropriate, with full traceability back to their origins.

- Modeling can be expressed with respect to entire community repositories, rather than within the confines of discrete models or templates. You might still create discrete models – clouds of metatexts – but those metatexts will be straightforwardly available for use in other models and less formally modeled contexts.
- You can express ideas and opinions about texts that are neither location- nor instance- specific. "Whenever you're drafting an XYZ agreement, be sure to cover topics A and B *somewhere*."
- Taxonomies can be bottom-up and folksonomic – and thus emergent and resilient, rather than top-down and brittle. People can label anything anyway they wish, although they are increasingly guided by the patterns that emerge.
- Drafting sessions can be guided by dynamically assembled collections of modeling statements, filtered as needed by user preferences and harmonized as necessary by automated routines.
- As drafters work they can consult dynamically assembled windows of metatexts that are associated directly or indirectly with the passages of text in draft that are currently in focus.
- Specific moves in a system-user drafting session can be modeled as standardized state transitions from prescriptive to descriptive metatext. For instance, when a user has adopted and chosen to process a passage containing a location as to which a field metatext has been associated ("Insert plaintiff name here"), her interaction with the system would result in the given answer ("Smith") being tagged with the metatext "Name of plaintiff." Similarly, a location starting out with an associated metatext that instructs one to include some passage IF a certain situation obtains becomes a passage that is the referent of a statement that it is there BECAUSE User X said that that situation obtained.
- Texts that have been composed within a drafting system of this sort will be far more richly described and thus more automatically re-processable.
- The network itself can be continually mined for collective knowledge, using techniques like the PageRank algorithm[4] for instance to compute the 'goodness' of particular precedents and metatexts.

[4] The is the primary algorithm behind the Google search engine. See http://en.wikipedia.org/wiki/PageRank

6. OBSERVATIONS

6.1 The advantages and challenges of digital documents

Nearly every legal and business document nowadays spends some time in electronic form, which usefully provides an unambiguous sequence of specific graphemes and a set of coordinates through which every location and region can be uniquely specified. But digital documents also can pose requirements to deal with features of a document above and beyond what you see on paper, such as what styles to apply where, what forms of automation to enable, and what metadata to include.

6.2 The subjectivity of satisfaction

Whether a document satisfies a model, or any part thereof, is often not machine-determinable, and involves the judgment of some human actor. Statements about satisfaction need to be regarded as defeasible claims subject to argumentation. Authors and reviewers may well come to different conclusions about what aspects of a document satisfy what aspects of a model it purports to follow.

If a model includes statements with intermediate values on the "spectrum of advisability" there are corresponding spectra of satisfaction. When it exhibits a merely mildly unadvised property, a document may still be regarded as satisfying a model.

Models that are communicated in the limited expressiveness of a document assembly program template[5] can be said to be satisfied when a document can be assembled from the template without editing.

6.3 The generality of document modeling

There's little distinctively legal about legal document modeling.

While document modeling as considered here is especially relevant to the complex documents often found in legal contexts, and employs deontological and argumentation constructs frequently encountered in the law-and-AI literature, there is little about it that is peculiarly applicable to legal documents. The fact that legal documents themselves often embody normative statements is of no particular moment (other than as a source of confusion when one conflates norms *of* documents with norms *in* documents.)

Of course the tasks of creating models of legal documents, and of drafting documents in ways that make use of such models, draw upon knowledge of the substantive law, of legal procedures, and of legal conventions with regard to style and form. Particular models will often be domain specific. The concepts and purposes expressed in a legal document model will usually be specifically legal. The models may rest on distinctively legal analysis, or require it for their application. But the phenomena of modeling is mostly domain-independent.

6.4 The need for greater expressiveness

A key motivation for the theoretical work we've undertaken is the manifest need for a more general and expressive framework

[5] Such templates can be regarded as consisting entirely of imperatives because, among other reasons, they do not support any clear way to communicate whether a particular 'boilerplate' passage is required to be included as is, where it is, or is merely permitted to be so left.

within which to represent the things that people say when they talk about documents. Most existing tools and methods are very incomplete in terms of the kinds of modeling statements they can represent or process. A comprehensive formalism should be rich enough to capture the full range of document modeling behavior encountered in the real world. Among other things, that means doing justice to the normative, argumentative, and 'messy' nature of such behavior. It includes being attentive to the ways in which 'substantive' and 'stylistic' considerations may interpenetrate, and in which strategic purposes may dwarf 'legal' ones.

7. CONCLUSION AND FURTHER WORK

We have sketched a model of document modeling that is basic but quite general. It encompasses deontic nuances and argumentative dimensions not previously given adequate attention. Our eventual formalisms can serve as ways to canonically represent the document models implicit in informal representation methods, and to characterize the gaps, contradictions, and ambiguities present in such models.

Next steps on the theoretical front include formalizing the processes by which model *satisfaction* can be described and debated, on the foundation of argumentation theory in philosophy, and the operations through which models and compliance descriptions can support drafting processes themselves.

On the practical front, we are exploring how formal languages for expressing legal knowledge and arguments, such as the Legal Knowledge Interchange Format (LKIF) [10], and legal reasoning and argumentation tools like the LKIF reference inference engine, Carneades[6], can be harnessed together with document assembly engines like HotDocs to provide a rich environment within which people can simultaneously reason about both texts and the purposes they are crafted to serve.

REFERENCES

[1] Bench-Capon, T., and Staniford, G. PLAID — proactive legal assistance. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law* (College Park, 1995), pp. 81–88.

[2] Biagioli, C., Francesconi, E., Spinosa, P., and Taddie, M. A legal drafting environment based on formal and semantic XML standards. In *Proceedings of the Tenth International Conference on Artificial Intelligence and Law*. Bologna, June 2005. pp. 244-245

[3] Boer, A., Winkels, R., and Vitali, F.. Proposed XML standards for law: Metalex and LKIF. In Arno R. Lodder and Laurens Mommers, editors, Legal Knowledge and Information Systems. Jurix 2007: The Twentieth Annual Conference Annual Conference, volume 165 of Frontiers in Artificial Intelligence and Applications, pages 19-28. IOS Press, December 2007.

[4] Branting, L. K. Techniques for Automated Judicial Document Drafting, *International Journal of Law & Information Technology*, 6(2):214-229 (1998).

[5] Branting, L.K., Lester, J., and Callaway, C. Automating Judicial Document Drafting: A Discourse-Based Approach, *Artificial Intelligence and Law*, 6(2-4):105-110 (1998)

[6] Branting, L.K., Callaway, C., Mott, B., and Lester, J. Integrating Discourse and Domain Knowledge for Document Drafting. Oslo, Norway, July 14-17, 1999

[7] Branting, L.K., Lester, J., and Callaway, C. Automated Drafting of Self-Explaining Legal Documents. *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, June 30-July 3, 1997, University of Melbourne, Melbourne, Australia, pp. 72-81

[8] Daskalopulu, A. and Sergot, M. 1995. A Constraint-Driven System for Contract Assembly. In *Proceedings of the 5th International Conference on Artificial Intelligence and Law (ICAIL-05)* (ACM Press), pp. 62-70.

[9] ESTRELLA Project. The legal knowledge interchange format (LKIF). Deliverable 4.3, European Commission, 2008.

[10] ESTRELLA Project. The reference LKIF inference engine. Deliverable 4.3, European Commission, 2008.

[11] ESTRELLA Project, Reference Open Source Legal Content Management System (CMS), Deliverable 3.4, European Project IST-2004-027655, 2008. [Available at http://www.estrellaproject.org/]

[12] Gordon, T. F. A Theory Construction Approach to Legal Document Assembly. *Expert Systems in Law*. Ed. Antonio A. Martino. North-Holland, Amsterdam, 1992. 211-25.

[13] Hafner, C. and Lauritsen, M. Extending the power of automated legal drafting technology JURIX 2008

[14] Kerrigan, S. and Law, Kincho. Logic-Based Regulation Compliance-Assistance. In *Proceedings of the Ninth International Conference on Artificial Intelligence and Law*. Edinburgh, June 2003, pp. 126-135.

[15] Kowalewski, D. L. and Schneeberger, J.; KOKON: Wissensbasierte Konfigurierung von Verträgen; GI-16. Jahrestagung; Springer-Verlag; Berlin; 1986.

[16] Lauritsen, M. Technology Report: Building Legal Practice Systems with Today's Commercial Authoring Tools. 1 *Artificial Intelligence and Law* 87-102 (1992)

[17] Lauritsen, M. Knowing Documents. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*. Amsterdam, June 1993.

[18] Lauritsen, M. Ontologies and Openness in Law Practice Automation. Workshop on "Legal Knowledge Systems in Action," Eighth International Conference on Artificial Intelligence and Law, St. Louis, Missouri. May 2001 [http://www.capstonepractice.com/OntoOpen.html]

[19] Lauritsen, M. 2007. Current Frontiers in Legal Drafting Systems. Working paper for a tutorial at the Eleventh International Conference on Artificial Intelligence and Law, Palo Alto, California. July 2007. [Available at http://www.capstonepractice.com/CurrentFrontiers.pdf]

[20] Lupo, C. and Batini, C. (2003). A federative approach to laws access by citizens: The "Norme in Rete" system. In Traunmüller, R., editor, Proceedings of Second International Conference Electronic Government (EGOV), pages 413–416. Springer-Verlag.

[21] Moens, M. Improving Access to Legal Information: How Drafting Systems Help. In Lodder and Oskamp eds.,

[6] http://carneades.berlios.de

Information Technology & Lawyers: Advanced technology in the legal domain, from challenges to daily routines (Springer, 2006), pp. 119-136

[22] Poulin, D., Huard, G, and Lavoie, A. The other formalization of Law: SGML modeling and tagging. *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, June 30-July 3, 1997, University of Melbourne, Melbourne, Australia, pp. 82-88

[23] Sergot, M. J., Sadi, F., Kowalski, R. , Kriwaczek, R. A. , Hammond, P., and Cory, H. T.; The British Nationality Act as a Logic Program; Communications of the ACM; Vol. 29; 1986.

[24] Sprowl, J. 1979. Automating the Legal Reasoning Process: A Computer that uses Regulations and Statutes to Draft Legal Documents. 1 Am. B. Found. Res. J. 1-8

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