Contract Templates And Contract Automation Using LaTeX *

Greg Kochansky†

August 20, 2013

Abstract

This overview describes how lawyers can use LaTeX (a free, sophisticated document preparation system) to create dynamic contract templates that include fill-in-the-blank form fields, automatic numbering and re-numbering of nested clauses, and a system for easily toggling particular clauses on and off or selecting between alternative versions of clauses. This workflow generates well-formatted PDF files that are safe and easy to share with colleagues, clients, and counterparties.

1 Getting started.

Learning LaTeX may seem a bit overwhelming for the uninitiated, but it is no more difficult than learning basic HTML, search syntax for Lexis and WestLaw, or even the dreaded *Bluebook*. There are rules. Some may seem a bit arbitrary (although, in this case, they are very well thought out). Bottom line: this is a bit of a challenge, but it is not brain surgery.

And, anyone with even a passing familiarity with some other flavor of "markup", whether it be HTML, XML, or even Markdown, will get the hang of LaTeX quickly. The syntax is very straightforward, and a Google search will yield answers to most questions within the first five search results. We are dealing with a very well-documented, very reliable markup language.

Also keep in mind that, as with any other document automation system, setting up the templates is the hardest part. Once a template for a particular type of contract has been created, the learning curve flattens out considerably for those who just want to use that template. The creator can add comprehensive annotations to a template file, and all the end user has to do is fill in some transaction-specific language. So, all a law firm or organization needs is one LaTeX "expert" on staff. Everyone else just has to fill in a few blanks and press a button to produce a final document.

†JD, Harvard Law School. AB in English, Princeton University. Licensed to practice law in the State of New York and the Commonwealth of Massachusetts. Based in Cambridge, Massachusetts, Greg can be reached at greg@greg-k.com and 646-465-4897.

2 Why on earth would anyone want to take this approach?

In short, to avoid inefficiency, errors, inconsistencies, Microsoft Word in general, Word's "track changes" function in particular, and metadata.

1. **Efficiency.** Contract assembly automates repetitive tasks. If you are generating many versions of a contract and only 15% of each document contains unique content, then you have an opportunity to automate 85% of the drafting process. The potential efficiency gain is enormous.
2. **Limiting errors.** You might think the typical "find-and-replace" strategy is up to the task, but it really isn't. How many times have you tried to adapt a document in Word using "find-and-replace", only to find, while reviewing the final product, that you missed a few spots? Even worse, what if you didn't catch them before circulating the draft. It would be beyond embarrassing to send a draft to a client containing names, addresses, or other information clearly left over from a prior engagement for a different client. And divulging such information to a third party might even be considered professional negligence.
3. **Consistency.** Sometimes you just want to change a clause or two in a "form" contract that you are adapting for a particular purpose. Instead of cobbling together clauses from a bunch of different sources, it makes much more sense to have the most common options available to you within a single template document. The workflow described here makes it very easy, for instance, to toggle between different dispute resolution clauses or mutual/unilateral indemnification provisions.
4. **Avoiding Microsoft Word.** A lot of lawyers do not enjoy using Microsoft Word. There are too many options scattered over far too many "ribbons". Fixing formatting issues gets in the way of drafting. And, typically, different people use Word styles and formatting in different ways, leading to inconsistent results across a given organization. LaTeX builds an impregnable wall between content and formatting. If you centralize your templates, then every single document will leave your office sporting a consistent, professional "look", not to mention easy-to-read typography that beats anything Word can produce.
5. **Stripping metadata.** Metadata is a dangerous thing. It can and often does lead to the exposure of information that ought to remain confidential. Sure, there is third-party software like WorkShare Protect that will strip metadata from Word documents. This functionality even exists in more recent versions of Word itself (though the feature is buried in a sub-menu). But why add an extra, easy-to-forget step to the process? PDFs are inherently cleaner and safer to circulate than are Word documents.
6. **Tracking changes.** Transactional attorneys tend to love Word's "Track Changes" feature, but it is a terrible idea to rely on it in a contract nego-

tiation. Whether by accident or on purpose, it is so easy for a new change to be accepted prematurely so that the other side misses the change completely. You can use DeltaView and other third-party applications as a safety measure, but you can accomplish the very same thing with (much safer) PDFs by using the latest version of Adobe Acrobat.

3 Acknowledgments

As I mentioned, a wide variety of online resources are available to anyone working with LaTeX. In this overview, I have linked to the sources that showed me how to create this template.

In terms of the overall approach, I am indebted to the groundbreaking "Legal Markdown" repository built and maintained by Casey Kuhlman on GitHub. Legal markdown uses Ruby and YAML, rather than LaTeX, to accomplish text substitution, conditional clauses, and automated clause numbering.

4 Meet your friendly LaTeX editor (free download, easy installation).

The only software you will need to duplicate this workflow is a text editor capable of converting LaTeX markup into a final PDF document. You have a variety of options. I am suggesting Texmate here because it is easy to install and use.

1. Download Texmate for Windows, MacOS, or Ubuntu.
2. Follow the applicable installation instructions.
3. Download the model template.
4. Locate the Texmate icon and start the program.
5. In the top menu, navigate to File, then Open. Navigate to wherever you downloaded the example template and open it.
6. The template is just an example, not to be used as a real-world contract, but you can alter the file as you see fit. To generate a PDF, return to the top menu and navigate to Tools, then Quick Build (or press the F1 key).
7. The PDF output should open automatically. Using the icons at the top right of the viewing window, you can print the document or open and save it using your standard PDF viewer.

5 The model template.

To demonstrate how contract automation works in LaTeX, this paper references a sample template, which you can view here. It is a bare-bones non-disclosure

agreement. I strongly advise against using it as-is. It merely demonstrates all the features in this workflow. The template contains annotations to help the user to better understand how it is structured. Also, here is a sample PDF based on the template.

6 Key features of this workflow.

A usable contract template should allow you to enter all transaction-specific information in one place, to toggle between those very few provisions that may vary from deal to deal, and to automate the numbering and nesting of paragraphs and sub-paragraphs.

1. **Transaction-specific information.** Toward the beginning of the template file, all the "variables" appear in a section that looks like this:

```
% Define text to be substituted into the contract.

\newcommand{\TitleText}{}
\newcommand{\PartyOneLegalName}{}
\newcommand{\PartyOneEntityType}{}
\newcommand{\PartyOneEntityOrgState}{}
\newcommand{\PartyOneAddressOne}{}
\newcommand{\PartyOneAddressTwo}{}
\newcommand{\PartyOneEmailAddress}{}
\newcommand{\PartyOneSignatoryFirstName}{}
\newcommand{\PartyOneSignatoryLastName}{}
\newcommand{\PartyOneSignatoryPosition}{}
\newcommand{\PartyTwoLegalName}{}
\newcommand{\PartyTwoEntityType}{}
\newcommand{\PartyTwoEntityOrgState}{}
\newcommand{\PartyTwoAddressOne}{}
\newcommand{\PartyTwoAddressTwo}{}
\newcommand{\PartyTwoEmailAddress}{}
\newcommand{\PartyTwoSignatoryFirstName}{}
\newcommand{\PartyTwoSignatoryLastName}{}
\newcommand{\PartyTwoSignatoryPosition}{}
\newcommand{\StateLaw}{}
\newcommand{\VenueCity}{}
\newcommand{\VenueState}{}
```

   Between the curly brackets at the end of each line, an end user of the finished template simply enters the information that applies to the current transaction. For example:

```
% Define text to be substituted into the contract.
```

```
\newcommand{\TitleText}{Non-Disclosure Agreement}
\newcommand{\PartyOneLegalName}{OldCorp, LLC}
\newcommand{\PartyOneEntityType}{Limited Liability Company}
\newcommand{\PartyOneEntityOrgState}{the State of New York}
\newcommand{\PartyOneAddressOne}{100 15th Avenue}
\newcommand{\PartyOneAddressTwo}{New York, NY 10000}
\newcommand{\PartyOneEmailAddress}{info@oldcorp.com}
\newcommand{\PartyOneSignatoryFirstName}{John}
\newcommand{\PartyOneSignatoryLastName}{Smith}
\newcommand{\PartyOneSignatoryPosition}{Member and Manager}
\newcommand{\PartyTwoLegalName}{NewCorp, Inc.}
\newcommand{\PartyTwoEntityType}{Corporation}
\newcommand{\PartyTwoEntityOrgState}{the State of Delaware}
\newcommand{\PartyTwoAddressOne}{1 14th Avenue}
\newcommand{\PartyTwoAddressTwo}{New York, NY 10000}
\newcommand{\PartyTwoEmailAddress}{info@newcorp.new}
\newcommand{\PartyTwoSignatoryFirstName}{Mary}
\newcommand{\PartyTwoSignatoryLastName}{Jones}
\newcommand{\PartyTwoSignatoryPosition}{President}
\newcommand{\StateLaw}{the State of New York}
\newcommand{\VenueCity}{New York}
\newcommand{\VenueState}{New York}
```

There are only a few simple formatting rules that apply. First, the variable names cannot contain any characters other than uppercase and lowercase letters: no numbers, no spaces, and no special characters. Second, when specifying the replacement text for each variable, you have a bit more freedom, but there are still some formatting rules to keep in mind. Spaces are acceptable, as are numbers and special characters. However, it is necessary to type certain special characters in a particular way so that the LaTeX interpreter does not read them as commands rather than as literal text. This process is called "escaping". Here are the ten special characters that need to be "escaped":

```
& % $ # _ { } ~ ^ \
```

In order for the processor to interpret these special characters literally, they have to be typed in particular way. The first seven characters must be preceded by a backslash:

```
\& \% \$ \# \_ \{ \}
```

The final three characters require the use of a special syntax, which begins with a backslash, followed by a descriptive verbal code.[1].

[1] Explanation here

```
\textasciitilde \textasciicircum \textbackslash
```

To call these variables in the body of the template, the template creator types in the variable name, as it appears in the header section. If the variable occurs before another word in the body of the text, then the variable must be enclosed in backslashes. If the variable appears directly before a punctuation mark in the body of the text, then a backslash only need be appended to the beginning of the variable name. Compare:

```
This \TitleText\ is by and between . . . .
```

With:

```
This \TitleText.
```

There are also some special"backslash codes" that can be used in the body of the document without being defined as a variable in the header. The most useful code for contract-drafting is almost certainly the one for specifying the current day's date. Again, this special verbal code is accompanied in the body text by either one or two backslashes, depending on whether it appears directly before a punctuation mark. Compare:

```
This agreement is dated \today.
```

With:

```
\today\ this agreement was signed.
```

In the final document, any of these variables will be converted to the appropriate text. For instance:

```
This Non-Disclosure Agreement is by and between . . . .
```

Or:

```
This agreement is dated August 20, 2013.
```

2. **Conditional clauses.** Sometimes a template creator may want the end user of the template to have the option of choosing between a couple of variations of a clause or toggling a particular clause "on" or "off". Dispute resolution clauses are a good example. In some cases, the user might want to include an arbitration clause – perhaps one endorsed by the American Arbitration Association, JAMS, or the ICC. In other cases, for whatever reason, one or both parties may be more comfortable with a conventional venue clause that requires court-based dispute resolution. In the example template we are discussing here, the header contains a section that includes two conditional clauses, one that toggles between a one-way and two-way NDA format, and one that alters the dispute resolution language:

```
% Set conditional clauses.
\newcommand{\OneWayOrTwoWay}{two-way}
\newcommand{\ArbitrationOrCourt}{court}
```

The same rules from the previous section also apply to naming these conditional clauses. We will focus on the dispute resolution clause:

```
\newcommand{\ArbitrationOrCourt}{court}
```

It probably makes sense to use a variable name that presents an either-or choice between two options. In this case, the variable name makes clear that the user can specify either "arbitration" or "court" to toggle between the two different clauses.

In order for this toggling to work properly, the very beginning of the template must include a particular "package" called "xstring", a small program that handles this sort of conditional processing. This line of code is clearly marked for you in the template file we are discussing:

```
% This package processes the conditional clauses
\usepackage{xstring}
```

In the body of the document, you need to place the two possible clauses in the appropriate location (in this case, as part of the boilerplate toward the end of the contract):

```
\IfSubStr{\ArbitrationOrCourt}{arbitration}{\item {\bf Dispute
Resolution.} Any controversy or claim arising out of or relating
to this Agreement, or the breach thereof, whether sounding in
contract, tort, or otherwise, shall be settled by arbitration
administered by the American Arbitration Association under its
Commercial Arbitration Rules, and judgment on the award rendered
by the arbitrator(s) may be entered in any court having jurisdiction
thereof.}{\item {\bf Dispute Resolution.} Any controversy or claim
arising out of or relating to this Agreement, or the breach thereof,
whether sounding in contract, tort, or otherwise, shall be decided
solely and exclusively by State or Federal courts located in
\VenueCity, \VenueState.}
```

This syntax is less complicated than it may seem:

```
\IfSubStr{[Variable Name With a backslash at the beginning.]}{[The
code word for selecting the first option.]}{[The text of the first
option, including all formatting marks.]}{[The text of the second
option, including all formatting marks.]}
```

Thus, if you want to be able to toggle between two clauses, you can accomplish this with a single line in the header. To toggle among three or more options for a particular clause, include a separate variable for each clause. In the body of the document, include conditional syntax for each variable, but leave the second option blank. With this approach, an end user can toggle one of those clauses "on" while leaving the other ones set to "off".[2].

3. **Automatic numbering.** The "bullets and numbering" functionality in Word can be maddening. It can interfere with your flow when drafting, and it can lead to structural errors in your draft. LaTeX offers a simple, powerful system for auto-numbering. All ordered/numbered lists take the form:

```
\begin{enumerate}
  \item First item.
  \item Second item.
  \item Third item.
\end{enumerate}
```

In the final document, the markup above will generate a list that looks like this:

1. First item.
2. Second item.
3. Third item.

You can nest sub-sections within any such section list:

```
\begin{enumerate}
  \item First item.
  \item Second item.
  \item Third item.
  \begin{enumerate}
    \item First item.
    \item Second item.
    \item Third item.
  \end{enumerate}
\end{enumerate}
```

In the final document, the markup above will generate a list that looks like this:

1. First item.

[2]Explanation here

2. Second item.
3. Third item.
   (a) First item.
   (b) Second item.
   (c) Third item.

   You can nest ordered lists four levels deep, which should be sufficient for contract-drafting.[3].

4. **Other formatting conventions.** The other main formatting convention to keep in mind is that conventional quotation marks will not produce "smart" or "curly" quotation marks in the final document. Instead, for open quotes, use either one backtick character (`, for single quotes) or two backtick characters (``, for double quotes). For closed quotes, you use either one apostrophe character (', for single quotes) or two apostrophe characters ('', for double quotes).

   As you will notice in the example PDF, certain text (Level-1 paragraph titles, in particular) appears in boldface type. To accomplish this effect, enclose text in curly brackets with the string `\bf` placed at the beginning, inside the opening curly bracket:

   ```
   {\bf This text will appear in bold.}
   ```

   Finally, remember to include spaces after variables, typeface settings, and other markup so that individual words do not run together in the final document. A bit of trial and error is probably the best way to get comfortable with these conventions quickly.

7 Taking things a step further.

This is just an introduction to the possibilities when using LaTeX for legal document automation. A wide array of packages and techniques are available to add functionality beyond what is described here. Some fruitful areas for development include:

1. **Clause libraries.** It is possible to incorporate external text strings into a LaTeX file, which makes possible sophisticated, centralized clause libraries for boilerplate and other frequently-used provisions.

2. **Pop-up definitions.** Some packages, including "cooltips", allow for the creation of "pop-up" windows that appear when a reader hovers over a particular word or phrase in the document. Using this functionality, it would be possible to include a "pop-up" definition for every instance of

[3] Explanation here

a defined term in the document. Then, anyone reviewing the PDF could reference the definition of any defined term without having to flip back to the definitions section of the contract.

3. **Customized formatting.** The template described here uses relatively "vanilla" formatting. Templates allow for considerable customization. Digging even deeper, it is possible to create brand new document styles, which give the creator complete control over all aspects of a document's formatting.

4. **Export to other formats.** In this use case, the markup is exported only in PDF format. Using Pandoc or other document processing utilities, it is possible to generate outputs in a variety of other formats (including the Word *.docx format and OpenOffice *.odt format).

8 Conclusion

This introduction barely scrapes the surface when it comes to legal document automation using LaTeX. Importantly, these automation techniques are not confined to contract-drafting. They can be applied to any type of legal document, including corporate formation documents; opinion letters; briefs, pleadings, and judicial opinions; and all manner of correspondence. When you consider how much repetitive language appears in legal documents, and the many problems that can arise in the typical, Word-based workflow, it is worth giving a bit more thought to an alternate approach like this one.

Thanks to the simplicity of the markup, the availability of powerful packages that extend functionality, and the wealth of excellent (and free) documentation, I believe that any lawyer or paralegal with the will to do so can construct a robust document automation system using this method.