Madoko: Scholarly Markdown in the Cloud

Daan Leijen[1]

1 **Microsoft Research**
daan@microsoft.com

1 Overview

Madoko is an online scholarly markdown processor available at madoko.net (see Figure 1). The main design goal of Madoko is to enable light-weight creation of high-quality scholarly and industrial documents for the web and print, while maintaining John Gruber's Markdown philosophy of simplicity and focus on plain text readability [1]. Markdown is great for writing prose, but for more serious it falls short in several areas. Madoko provides many essential additions for more complex academic documents, including support for cross-references, mathematics, citations, bibliographies, figures, tables, code highlighting, setting code in proportional fonts (à la lhs2tex [4]), footnotes, etc.

Instead of a plethora of backends, Madoko concentrates on generating high quality HTML and PDF through LaTeX and BibTeX. There has been a lot of effort in Madoko to make the LaTeX generation robust and customizable while integrating well with the various academic document- and bibliography styles. Of course, this article itself was written in Madoko, and the HTML version can be viewed at http://tinyurl.com/n6k3kht [3].

Madoko documents are user programmable and we can specify rules to selectively transform or style document elements. This makes it easy to add custom domain specific blocks, like *exercise* or *answer*, that can be transformed, numbered, styled, etc. Transformations can be specified using regular expressions and standard CSS attributes.

Finally, the online version integrates seamlessly with Dropbox, GitHub, and OneDrive, making documents available anywhere on any device. Madoko synchronizes automatically and multiple authors can work concurrently on the same document. Madoko uses robust three-way merges on concurrent updates. This means that updates by others are not quite real-time as in other collaborative environments (although they are performed frequently), but anyone can now work off-line and still reliably merge when connecting again. Madoko.net is itself a HTML5 web application and the editor continues to work in the browser even when offline. Of course, you can always use the plain command line version of Madoko locally (`npm install -g madoko`).

Madoko is written in Koka – a function oriented language with (side) effect-inference [2]. The effect system proved essential to create a program that could reliably execute on both the server and client. This is how the online version of Madoko is able to render Madoko documents fully on the client side (and only using the server to render LaTeX math formulas).

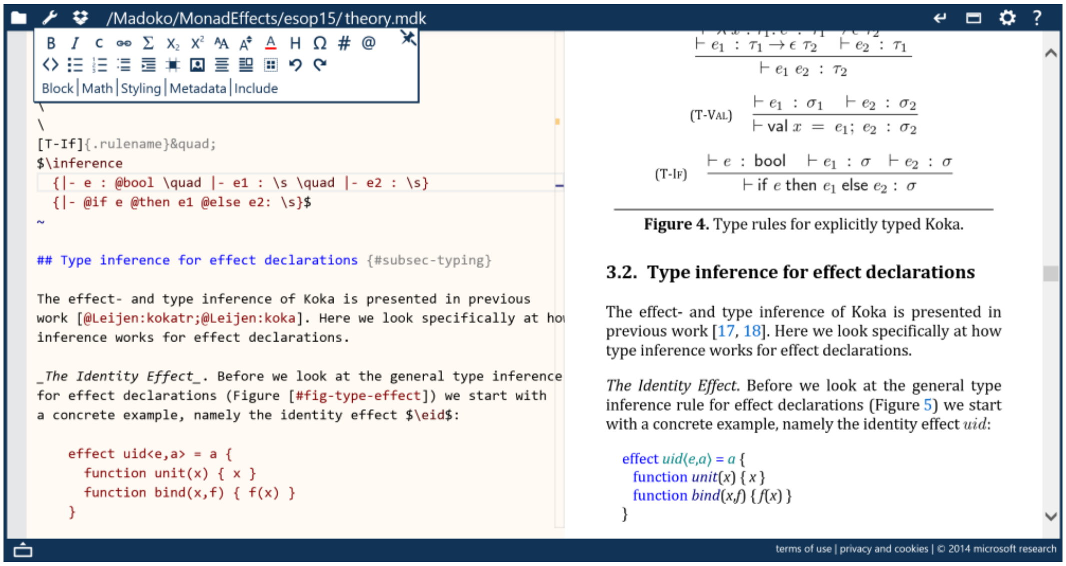
2 Style rules

As an example of the programmability of Madoko, we show how we can make a custom *callout* block. In the Madoko editor, we would like to write:

```
~ Callout
The formula $e=mc^2$ is famous.
~
```

The tildes are used in Madoko to delimit custom blocks where we try to minimize any needed markup. Similarly, the $ is used to delimit LaTeX formulas. The callout block is shown as:

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



/Madoko/MonadEffects/esop15/ theory.mdk
B I C Σ X₂ X² H Ω # @
Block | Math | Styling | Metadata | Include
terms of use | privacy and cookies | © 2014 microsoft research

```
\
[T-If]{.rulename}&quad;
$\inference
  {|- e : @bool \quad |- e1 : \s \quad |- e2 : \s}
  {|- @if e @then e1 @else e2: \s}$
~

## Type inference for effect declarations {#subsec-typing}

The effect- and type inference of Koka is presented in previous
work [@Leijen:kokatr;@Leijen:koka]. Here we look specifically at ho
inference works for effect declarations.

_The Identity Effect_. Before we look at the general type inference
for effect declarations (Figure [#fig-type-effect]) we start with
a concrete example, namely the identity effect $\eid$:

    effect uid<e,a> = a {
      function unit(x) { x }
      function bind(x,f) { f(x) }
    }
```

$$\frac{\vdash e_1 \;:\; \tau_1 \rightarrow \epsilon\; \tau_2 \qquad \vdash e_2 \;:\; \tau_1}{\vdash e_1\; e_2 \;:\; \tau_2}$$

$$(\text{T-VAL})\quad \frac{\vdash e_1 \;:\; \sigma_1 \qquad \vdash e_2 \;:\; \sigma_2}{\vdash \mathsf{val}\; x \;=\; e_1;\; e_2 \;:\; \sigma_2}$$

$$(\text{T-IF})\quad \frac{\vdash e \;:\; \mathsf{bool} \qquad \vdash e_1 \;:\; \sigma \qquad \vdash e_2 \;:\; \sigma}{\vdash \mathsf{if}\; e\; \mathsf{then}\; e_1\; \mathsf{else}\; e_2 \;:\; \sigma}$$

Figure 4. Type rules for explicitly typed Koka.

3.2. Type inference for effect declarations

The effect- and type inference of Koka is presented in previous work [17, 18]. Here we look specifically at how type inference works for effect declarations.

The Identity Effect. Before we look at the general type inference rule for effect declarations (Figure 5) we start with a concrete example, namely the identity effect uid:

```
effect uid⟨e,a⟩ = a {
  function unit(x) { x }
  function bind(x,f) { f(x) }
}
```

Figure 1 A screenshot of an article in the Madoko.net environment.

Note: The formula $e = mc^2$ is famous.

We can now freely use the callout blocks and concentrate on writing prose. The styling is done later using a metadata rule where we can use standard CSS attributes:

```
~Callout: margin-left=1em background-color=#FFC966 before="**Note**: "
          padding=0.5ex
```

This makes it very easy to style the final output while still being able to write the content in pleasant Markdown. It is also nice to be able to leverage standard CSS attributes without needing to know the plethora of LaTeX commands to achieve a similar effect. The `before` attribute is an example of the rewriting abilities of Madoko where we insert some text before the content. In general, arbitrary regular expression transformations can be applied.

3 Conclusion

Try Madoko at madoko.net. Madoko is still a young project and any feedback is much appreciated. We hope that we will see more and more academic articles that also come with a high quality HTML version – being able to resize and reflow makes it much more pleasant to read articles on a screen or tablet.

References

[1] John Gruber. Markdown. 2004. URL http://daringfireball.net/projects/markdown.

[2] Daan Leijen. Koka: Programming with row polymorphic effect types. In *MSFP*, 2014. doi:10.4204/EPTCS.153.8.

[3] Daan Leijen. Madoko: Scholarly markdown in the cloud. In *SNAPL'15: The Inaugural Summit on Advances in Programming Languages*, May 2015. HTML version available at: http://tinyurl.com/n6k3kht.

[4] Andres Löh. lhs2tex. 2004. URL http://www.andres-loeh.de/lhs2tex.

Created with Madoko.net.