Code Sprint Improved Approach for inserting Agda Code into LaTeX

Anton Setzer
Agda Implementors Meeting XXXVIII

13 May 2024

Agda

Implementors' Meeting

Swansea University

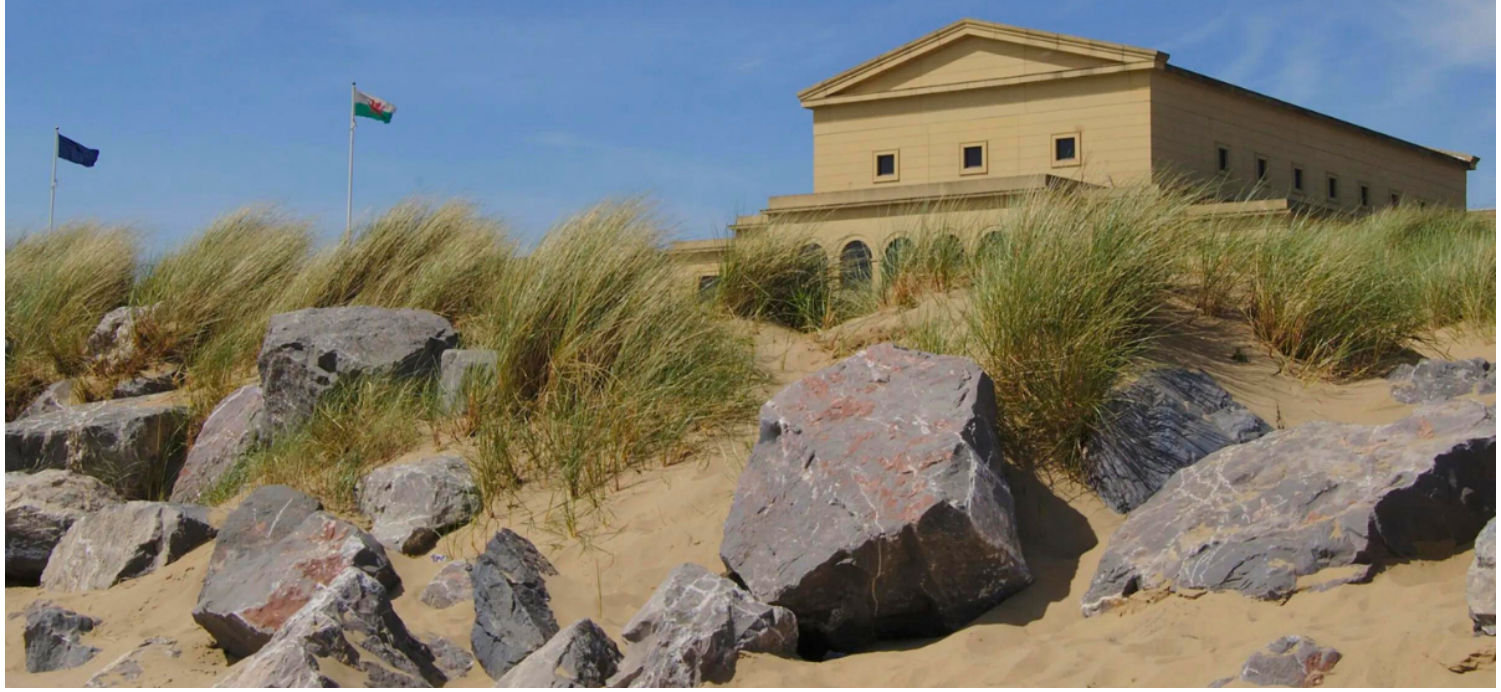
Prifysgol Abertawe

13-18 May 2024

Swansea University

Bay Campus

More information:
https://wiki.portal.chalmers.se/agda/Main/AIMXXXVIII



Code Sprint Improved Approach for inserting Agda Code into LaTeX

- Framework developed jointly by Andreas Abel, Stephan Adelsberger, AS
- Allows to easily incorporate Agda into LaTeX.

Example Code

Here is an example:

```
data Term : Set where
  _·_ : (c : Combinator)(tl : TermList) → Term
- _ denote infix arguments
```

Here is another example:

```
data NF⊎Red : Set where
  nf  : (t : Term) → NF⊎Red
  red : (t : Term) → NF⊎Red
```

Source Code Latex

```
\begin{frame}
\frametitle{Example Code}

Here is an example:

\extendedPredicativeMahloVersSeventerm

Here is another example:
\extendedPredicativeMahloVersSevennfred
end{frame}
```

(backslash before `end` removed to get this slide through LaTeX)

Screenshot of LaTeXcode

File Edit Options Buffers Tools TeX Text Help

Save Undo

```
\begin{frame}
\frametitle{Example Code}
Here is an example:

\extendedPredicativeMahloVersSeventerm

Here is another example:

\extendedPredicativeMahloVersSevennfred

\end{frame}
```

-:**- main.tex<codesprintAgdaLatex> 56% L30 (LaTeX Abbrev) 18:06 1.53

Agda Source Code with Tags Part 1

File Edit Options Buffers Tools Agda Help

Save Undo

```
--@PREFIX@extendedPredicativeMahloVersSeven

{-# OPTIONS --no-positivity-check #-}

module extendedPredicativeMahloVers7 where

open import eqReasoning
open import Agda.Builtin.Equality

data Bool : Set where
  true false : Bool

data ⊥ : Set where

record ⊤  : Set where
  constructor tt

atom : Bool → Set
atom true = ⊤
atom false = ⊥


data _∧_ (A B : Set) : Set where
```

∏U:--- extendedPredicativeMahloVers7.agda<codesprintAg

Agda Source Code with Tags Part 2

File Edit Options Buffers Tools Agda Help

Save Undo

```
mutual
--@BEGIN@term
  data Term : Set where
    _··_ : (c : Combinator)(tl : TermList) → Term
  -- _ denote infix arguments
--@END

  data TermList : Set where
    []  : TermList
    _::_ : (t : Term)(tl : TermList) → TermList

-- <_,_> : Term → Term → Term
-- < r , s > = πc ·· (r :: s :: [])

pattern nat  = natc ·· []
pattern id  = idc ·· []
pattern ∅  = ∅c ·· []
pattern mahlo = mahloc  ·· []
pattern kc r s l = kcc ·· (r :: s :: l )
pattern sc r s t l = scc ··  (r :: s :: t :: l)
pattern π r s = πc ·· ( r :: s :: [])
```

-U:--- extendedPredicativeMahloVers7.agda<codesprintAgda

Agda Source Code with Tags Part 3

File Edit Options Buffers Tools Agda Help

Save Undo

```
_·L_ : Term → TermList → Term
t ·L [] = t
t ·L (s :: sl) = (t · s) ·L sl


--@BEGIN@nfred
data NF⊎Red : Set where
  nf   : (t : Term) → NF⊎Red
  red  : (t : Term) → NF⊎Red
--@END

data NF⊎RedList : Set where
  nfl  : (tl : TermList) → NF⊎RedList
  redl : (tl : TermList) → NF⊎RedList




_·nfl_ : Combinator → NF⊎RedList → NF⊎Red
c ·nfl (nfl tl) = nf (c ·· tl)
c ·nfl (redl tl) = red (c ·· tl)
```

∏U:--- extendedPredicativeMahloVers7.agda<codesprintAgda

Overview

- Essential idea
  - **Add tags to Agda code** which give names to parts of the Agda code you want to incorporate into the LaTeX document.
  - Generate **lagda files** from the Agda code.
  - Generate **LaTeX code** from the lagda files.
  - For part of the code tagged with `name`, the LaTeX code has a `\newtheorem{\name}{..}` with body being the LaTeX version of the Agda code in question.
  - In addition we add a global **PREFIX** added in front of all tags.
    - Useful since definitions with the same name occur in different Agda files (e.g. different versions of the same code).
- Possibility to create as well inline LaTeX versions of **Agda code**.
- Could probably moved to the new version of creating latex code without type checking
  - Sometimes running Agda is type consuming because you start for each lagda file from scratch.
  - Mechanism to optionally guarantee type checking of original Agda code needed.

Machinery

- We have a Make file which needs to be customised to
  - define the Agda files to create lagda files from
  - define the latex files to run.
- A few **sed** scripts which convert the text into lagda and slightly tweak it.
- Mechanism for removing tags could be easily created (lost some code for it).

Advantages/Disadvantages

- Advantages
  - Works directly on your production Agda code
  - Avoids problems of working in different modes (LaTeX vs Agda mode)
  - Agda code is fully type checked.
    - **Statement in papers**
      All of the Agda code shown in this paper was derived from the type-checked Agda code.
  - Once set up very robust (adapts easily to changes of Agda code).
- Disadvantages
  - Machinery with Makefile a bit handcrafted.
  - Running Agda sometimes time consuming.
    - Could be fixed by using `agda --latex`

Code Sprint

- Document the approach.
- Maybe adapt it to using `agda --latex`
  - But optionally enforce type checking of all original Agda files used.
- Hopefully one day some variant of this becomes part of Agda