# Correct-by-Construction Pretty-Printing

Nils Anders Danielsson

IFIP WG 2.1 Meeting #70,
Schloss Reisensburg, 2013-07-04

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° 247219.

# Pretty-printing

$$\text{add}\ (\text{mul}\ 1\ 2)\ (\text{mul}\ 3\ (\text{add}\ 4\ 5))$$

$$\Downarrow$$

```
1 * 2 + 3 * (4 + 5)
```

```
1 * 2 +
3 * (4 + 5)
```

# Classical pretty-printing combinators

Implemented by user:

$$pretty \;:\; A \;\rightarrow\; Doc$$

Combinator interface:

$$\begin{aligned}
&Doc &&: Set \\
&render &&: \mathbb{N} \rightarrow Doc \rightarrow String \\
&\textsf{text} &&: String \rightarrow Doc \\
&_\diamond_ &&: Doc \rightarrow Doc \rightarrow Doc \\
&\textsf{line} &&: Doc \\
&\vdots
\end{aligned}$$

# Correctness

Assume that we have a parser:

$$parse : String \rightarrow List\ A$$

Round-tripping property:

$$\forall\ w\ x \rightarrow x \in parse\ (render\ w\ (pretty\ x))$$

How can this property be proved?

# This work

I use types to ensure that the round-tripping property holds by construction.

# Documents

Before:

$$Doc : Set$$

Now:

$$\begin{aligned} Grammar &: Set \rightarrow Set \\ Doc &: Grammar\ A \rightarrow A \rightarrow Set \end{aligned}$$

# Grammars

Relational semantics for grammars:

$$\_\in\_\cdot\_ \ : \ A \ \rightarrow \ Grammar\ A \ \rightarrow \ String \ \rightarrow \ Set$$

$x \ \in \ g \ \cdot \ s$ means that the string $s$ and value $x$ are generated by $g$.

# Pretty-printers

Before:

$$pretty : A \to Doc$$

Now:

$$\begin{aligned} g &: Grammar\ A \\ pretty &: (x : A) \to Doc\ g\ x \end{aligned}$$

# Renderers

Before:

$$render : \mathbb{N} \to Doc \to String$$

Now:

$$\begin{aligned} render &: \mathbb{N} \to Doc\ g\ x \to String \\ parsable &: \forall\ w\ (d : Doc\ g\ x) \to \\ &\qquad x \in g \cdot render\ w\ d \end{aligned}$$

Correctness (by construction):

$$\forall\ w\ x \to x \in g \cdot render\ w\ (pretty\ x)$$

No guarantee of "prettiness".

# Overview

In talk:

- Grammars.
- Documents.
- Simple examples.

Not in talk:

- Renderer (based on Wadler's).

# Grammars

For simplicity: regular expressions.

$$
\begin{array}{lll}
\multicolumn{3}{l}{\textbf{data}\ \mathit{Grammar} : \mathit{Set} \to \mathit{Set}_1\ \textbf{where}} \\
\quad \emptyset & : & \mathit{Grammar}\ A \\
\quad \varepsilon & : & A \to \mathit{Grammar}\ A \\
\quad \mathsf{char} & : & \mathit{Char} \to \mathit{Grammar}\ \mathit{Char} \\
\quad \_\circledast\_ & : & \mathit{Grammar}\ (A \to B) \to \mathit{Grammar}\ A \to \\
 & & \mathit{Grammar}\ B \\
\quad \_|\_ & : & \mathit{Grammar}\ A \to \mathit{Grammar}\ A \to \\
 & & \mathit{Grammar}\ A \\
\quad \_\star & : & \mathit{Grammar}\ A \to \mathit{Grammar}\ (\mathit{List}\ A)
\end{array}
$$

# Semantics of grammars

$$\frac{}{x \in \varepsilon\, x \cdot [\,]} \qquad \frac{}{c \in \mathsf{char}\, c \cdot [c]}$$

$$\frac{f \in g_1 \cdot s_1 \qquad x \in g_2 \cdot s_2}{f\, x \in g_1 \circledast g_2 \cdot s_1 +\!\!+\, s_2}$$

$$\frac{x \in g_1 \cdot s}{x \in g_1 \mid g_2 \cdot s} \qquad \frac{x \in g_2 \cdot s}{x \in g_1 \mid g_2 \cdot s}$$

$$\frac{xs \in \varepsilon\, [\,] \mid \varepsilon\, \_{::}\_ \circledast g \circledast g \star \cdot s}{xs \in g \star \cdot s}$$

# Semantics of grammars

$$\frac{}{x \in \varepsilon\, x \cdot [\,]} \qquad \frac{}{c \in \mathsf{char}\; c \cdot [c]}$$

$$\frac{f \in g_1 \cdot s_1 \qquad x \in g_2 \cdot s_2}{f\, x \in g_1 \circledast g_2 \cdot s_1 +\!\!+\, s_2}$$

$$\frac{x \in g_1 \cdot s}{x \in g_1 \mid g_2 \cdot s} \qquad \frac{x \in g_2 \cdot s}{x \in g_1 \mid g_2 \cdot s}$$

$$\frac{}{[\,] \in g \star \cdot [\,]} \qquad \frac{x \in g \cdot s_1 \qquad xs \in g \star \cdot s_2}{x :: xs \in g \star \cdot s_1 +\!\!+\, s_2}$$

# Some grammar combinators

$$\_\lessdot\!\circledast\_ \; : \; Grammar\ A \;\rightarrow\; Grammar\ B \;\rightarrow\; Grammar\ A$$

$$g_1 \lessdot\!\circledast\, g_2 \;=\; \varepsilon\ (\lambda\ x\ \_\ \rightarrow\ x) \circledast g_1 \circledast g_2$$

$$\_+ \; : \; Grammar\ A \;\rightarrow\; Grammar\ (List\ A)$$

$$g + \;=\; \varepsilon\ \_{::}\_ \circledast g \circledast g \star$$

# Some grammar combinators

$$whitespace : Grammar\ Char$$
$$whitespace = \textsf{char}\ \texttt{' '} \mid \textsf{char}\ \texttt{'\textbackslash n'}$$

$$string : String \to Grammar\ String$$
$$string\ [\,] = \varepsilon\ [\,]$$
$$string\ (c :: s) = \varepsilon\ \_::\_ \circledast \textsf{char}\ c \circledast string\ s$$

# Documents

$$
\begin{array}{lll}
\textbf{data}\ \mathit{Doc} & : & \mathit{Grammar}\ A \to A \to \mathit{Set}_1\ \textbf{where} \\
\_\diamond\_ & : & \mathit{Doc}\ g_1\ f \to \mathit{Doc}\ g_2\ x \to \\
 & & \mathit{Doc}\ (g_1 \circledast g_2)\ (f\ x) \\
\mathsf{text} & : & \mathit{Doc}\ (\mathit{string}\ s)\ s \\
\mathsf{line} & : & \mathit{Doc}\ (\varepsilon\ \mathsf{unit} \lessdot\!\circledast\ \mathit{whitespace}\ +)\ \mathsf{unit} \\
\mathsf{nest} & : & \mathbb{N} \to \mathit{Doc}\ g\ x \to \mathit{Doc}\ g\ x \\
\mathsf{group} & : & \mathit{Doc}\ g\ x \to \mathit{Doc}\ g\ x \\
\mathsf{embed} & : & (\forall\ s \to x_1 \in g_1 \cdot s \to x_2 \in g_2 \cdot s) \to \\
 & & \mathit{Doc}\ g_1\ x_1 \to \mathit{Doc}\ g_2\ x_2
\end{array}
$$

# Defined document combinators

To handle $\_|\_$:

$$\begin{aligned} & left : Doc\ g_1\ x \rightarrow Doc\ (g_1 \mid g_2)\ x \\ & left\ d = \text{embed} \ldots d \end{aligned}$$

$$\begin{aligned} & right : Doc\ g_2\ x \rightarrow Doc\ (g_1 \mid g_2)\ x \\ & right\ d = \text{embed} \ldots d \end{aligned}$$

Embedding proofs:

$$\forall\ s \rightarrow x \in g_1 \cdot s \rightarrow x \in g_1 \mid g_2 \cdot s$$

$$\forall\ s \rightarrow x \in g_2 \cdot s \rightarrow x \in g_1 \mid g_2 \cdot s$$

# Defined document combinators

To handle $\varepsilon$:

$$\begin{aligned} empty &: Doc\ (\varepsilon\ x)\ x \\ empty &= \mathsf{embed}\ \ldots\ (\mathsf{text}\ \{s\ =\ \texttt{""}\}) \end{aligned}$$

Embedding proof:

$$\begin{aligned} \forall\ s\ \rightarrow\ \texttt{""}\ &\in\ string\ \texttt{""}\ \cdot\ s\ \rightarrow \\ x\ &\in\ \varepsilon\ x\ \qquad\quad \cdot\ s \end{aligned}$$

# Defined document combinators

To handle $\_ \lessdot\!\circledast \_$:

$$
\begin{aligned}
\_ \lessdot\!\diamond \_ \; &: \; Doc\; g_1\; x \;\rightarrow\; Doc\; g_2\; y \;\rightarrow \\
&\phantom{:}\;\; Doc\; (g_1 \lessdot\!\circledast\; g_2)\; x \\
d_1 \lessdot\!\diamond\; d_2 \;&=\; empty \diamond d_1 \diamond d_2
\end{aligned}
$$

Recall that $g_1 \lessdot\!\circledast\; g_2 \;=\; \varepsilon\; (\lambda\, x\; \_ \;\rightarrow\; x) \circledast g_1 \circledast g_2$.

# Simple example

$$
\begin{aligned}
&bit \;:\; Grammar\ Bool\\
&bit \;=\; \varepsilon\ \mathsf{true} \circledast\!\!< string\ \texttt{"1"}\\
&\quad\;\mid\; \varepsilon\ \mathsf{false} \circledast\!\!< string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} \;:\; (b : Bool) \to Doc\ bit\ b\\
&bit_{\mathsf{P}}\ b \;=\; ? \quad \texttt{--}\ Doc\ bit\ b
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit \;:\; Grammar\ Bool\\
&bit \;=\; \varepsilon\ \mathsf{true}\ \lessdot\!\circledast\ string\ \texttt{"1"}\\
&\qquad \mid\ \varepsilon\ \mathsf{false}\ \lessdot\!\circledast\ string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} \;:\; (b \;:\; Bool) \;\to\; Doc\ bit\ b\\
&bit_{\mathsf{P}}\ \mathsf{true} \;=\; ? \quad \texttt{--}\ Doc\ bit\ \mathsf{true}\\
&bit_{\mathsf{P}}\ \mathsf{false} \;=\; ? \quad \texttt{--}\ Doc\ bit\ \mathsf{false}
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit : Grammar\ Bool \\
&bit = \varepsilon\ \mathsf{true} \circledast\ string\ \texttt{"1"} \\
&\quad\mid\ \varepsilon\ \mathsf{false} \circledast\ string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} : (b : Bool) \to Doc\ bit\ b \\
&bit_{\mathsf{P}}\ \mathsf{true} = left\ ? \quad \text{-- } Doc\ (\varepsilon\ \mathsf{true} \circledast\ string\ \texttt{"1"}) \\
&\qquad\qquad\qquad\qquad\ \text{-- } \qquad\quad \mathsf{true} \\
&bit_{\mathsf{P}}\ \mathsf{false} = ? \qquad\quad \text{-- } Doc\ bit\ \mathsf{false}
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit \;:\; Grammar\ Bool \\
&bit \;=\; \varepsilon\ \mathsf{true} \;\lessdot\!\circledast\; string\ \texttt{"1"} \\
&\qquad \mid\; \varepsilon\ \mathsf{false} \;\lessdot\!\circledast\; string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} \;:\; (b \;:\; Bool) \;\to\; Doc\ bit\ b \\
&bit_{\mathsf{P}}\ \mathsf{true} \;=\; left\ (? \lessdot\!\diamond\ ?) \qquad \text{-- } Doc\ (\varepsilon\ \mathsf{true})\ \mathsf{true} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\;\; \text{-- } Doc\ (string\ \texttt{"1"})\ s \\
&bit_{\mathsf{P}}\ \mathsf{false} \;=\; ? \qquad\qquad\qquad\; \text{-- } Doc\ bit\ \mathsf{false}
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit : Grammar\ Bool \\
&bit = \varepsilon\ \mathsf{true} \circledast\!\!\!< string\ \texttt{"1"} \\
&\quad\ \mid\ \varepsilon\ \mathsf{false} \circledast\!\!\!< string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} : (b : Bool) \rightarrow Doc\ bit\ b \\
&bit_{\mathsf{P}}\ \mathsf{true} = left\ (empty \diamond\!\!< ?) \quad \texttt{--}\ Doc\ (string\ \texttt{"1"})\ s \\
&bit_{\mathsf{P}}\ \mathsf{false} = ? \quad \texttt{--}\ Doc\ bit\ \mathsf{false}
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&\mathit{bit} : \mathit{Grammar}\ \mathit{Bool} \\
&\mathit{bit} = \varepsilon\ \mathsf{true} \circledast\!\!\lessdot\ \mathit{string}\ \texttt{"1"} \\
&\quad\ \ \mid\ \varepsilon\ \mathsf{false} \circledast\!\!\lessdot\ \mathit{string}\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{bit}_{\mathsf{P}} : (b : \mathit{Bool}) \to \mathit{Doc}\ \mathit{bit}\ b \\
&\mathit{bit}_{\mathsf{P}}\ \mathsf{true} = \mathit{left}\ (\mathit{empty} \diamond\!\!\lessdot\ \mathsf{text}) \\
&\mathit{bit}_{\mathsf{P}}\ \mathsf{false} = ? \qquad \text{-- } \mathit{Doc}\ \mathit{bit}\ \mathsf{false}
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit \;:\; Grammar\ Bool \\
&bit \;=\; \varepsilon\ \mathsf{true}\ \lessdot\!\circledast\ string\ \texttt{"1"} \\
&\qquad\mid\ \varepsilon\ \mathsf{false}\ \lessdot\!\circledast\ string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} \;:\; (b \;:\; Bool) \;\rightarrow\; Doc\ bit\ b \\
&bit_{\mathsf{P}}\ \mathsf{true} \;=\; left\ \ (empty \lessdot\!\diamond\ \mathsf{text}) \\
&bit_{\mathsf{P}}\ \mathsf{false} \;=\; right\ (empty \lessdot\!\diamond\ \mathsf{text})
\end{aligned}
$$

# Simple example

$$
\begin{aligned}
&bit \;:\; Grammar\ Bool\\
&bit \;=\; \varepsilon\ \mathsf{true} \lessdot\!\circledast\ string\ \texttt{"1"}\\
&\quad\;\;\mid\; \varepsilon\ \mathsf{false} \lessdot\!\circledast\ string\ \texttt{"0"}
\end{aligned}
$$

$$
\begin{aligned}
&bit_{\mathsf{P}} \;:\; (b : Bool) \to Doc\ bit\ b\\
&bit_{\mathsf{P}}\ \mathsf{true} \;=\; left\ \ (empty \lessdot\!\diamond\ \mathsf{text})\\
&bit_{\mathsf{P}}\ \mathsf{false} \;=\; right\ (empty \lessdot\!\diamond\ \mathsf{text})
\end{aligned}
$$

$render\ 10\ (bit_{\mathsf{P}}\ \mathsf{false}) \equiv \texttt{"0"}$ $\qquad$ $\mathsf{false} \in bit \cdot \texttt{"0"}$

$render\ 1\ (bit_{\mathsf{P}}\ \mathsf{true}) \equiv \texttt{"1"}$ $\qquad$ $\mathsf{true} \in bit \cdot \texttt{"1"}$

# More defined document combinators

To handle $\_\star$:

$$nil \; : \; Doc\,(g\,\star)\,[\,]$$
$$nil \; = \; \textsf{embed} \; ... \; empty$$

Embedding proof:

$$\forall\; s \;\to\; [\,] \;\in\; \varepsilon\,[\,] \;\cdot\; s \;\to$$
$$[\,] \;\in\; g\,\star \;\cdot\; s$$

# More defined document combinators

To handle $\_\star$:

$$
\begin{aligned}
cons \;:\; & Doc\; g\; x \;\to\; Doc\; (g \star)\; xs \;\to \\
& Doc\; (g \star)\; (x :: xs) \\
cons\; d_1\; d_2 \;=\; & \textsf{embed} \;\ldots\; (empty \diamond d_1 \diamond d_2)
\end{aligned}
$$

Embedding proof:

$$
\begin{aligned}
\forall\; s \;\to\; & x :: xs \;\in\; \varepsilon\; \_::\_ \circledast g \circledast g \star \;\cdot\; s \;\to \\
& x :: xs \;\in\; g \star \;\cdot\; s
\end{aligned}
$$

# Swallowing trailing whitespace

$$symbol : String \rightarrow Grammar\ String$$
$$symbol\ s = string\ s \circledast whitespace \star$$

$$symbol\text{-}nil : Doc\ (symbol\ s)\ s$$
$$symbol\text{-}nil = \mathsf{text} \diamond nil$$

$$symbol\text{-}line : Doc\ (symbol\ s)\ s$$
$$symbol\text{-}line = \mathsf{embed}\ \ldots\ (\mathsf{text} \diamond \mathsf{line})$$

Embedding proof:

$$\forall\ s' \rightarrow$$
$$s \in string\ s \circledast (\varepsilon\ \mathsf{unit} \circledast whitespace +) \cdot s' \rightarrow$$
$$s \in string\ s \circledast whitespace \star \cdot s'$$

# Pattern

- For (almost) every grammar combinator: one or more document combinators.
- Embedding proofs in reusable combinators, ideally not in pretty-printers.

# Another example

(Based on an example due to Doaitse and Olaf.)

$$\mathit{bit\text{-}list} : \mathit{Grammar}\ (\mathit{List}\ \mathit{Bool})$$
$$\mathit{bit\text{-}list} = (\mathit{bit} \lessdot\!\circledast\ \mathit{symbol}\ \texttt{";"})\ \star$$

# Another example

$$\mathit{bit\text{-}list} : \mathit{Grammar}\ (\mathit{List}\ \mathit{Bool})$$
$$\mathit{bit\text{-}list} = (\mathit{bit} \lessdot\!\circledast\ \mathit{symbol}\ \texttt{";"})\ \star$$

```
"1; 0; 0;"
"1;      0;0;\n    "
"1;\n\n\n      \n  0;    0;"
```

# Another example

$$\mathit{bit\text{-}list} : \mathit{Grammar}\ (\mathit{List}\ \mathit{Bool})$$
$$\mathit{bit\text{-}list} = (\mathit{bit} \circledast\!\!\lessdot\ \mathit{symbol}\ \texttt{";"})\ \star$$

$$\mathit{bit\text{-}list}_{\mathsf{P}} : (\mathit{bs} : \mathit{List}\ \mathit{Bool}) \to \mathit{Doc}\ \mathit{bit\text{-}list}\ \mathit{bs}$$
$$\mathit{bit\text{-}list}_{\mathsf{P}}\ [\,] = \mathit{nil}$$
$$\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: [\,]) = \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \lessdot\!\diamond\ \mathit{symbol\text{-}nil})\ \mathit{nil}$$
$$\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: \mathit{bs}) =$$
$$\quad \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \lessdot\!\diamond\ \mathsf{group}\ (\mathsf{nest}\ 1\ \mathit{symbol\text{-}line}))$$
$$\qquad\quad (\mathit{bit\text{-}list}_{\mathsf{P}}\ \mathit{bs})$$

# Another example

$$
\begin{aligned}
&\mathit{bit\text{-}list}_{\mathsf{P}} : (\mathit{bs} : \mathit{List}\ \mathit{Bool}) \rightarrow \mathit{Doc}\ \mathit{bit\text{-}list}\ \mathit{bs} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ [\,] = \mathit{nil} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: [\,]) = \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \diamond \mathit{symbol\text{-}nil})\ \mathit{nil} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: \mathit{bs}) = \\
&\quad \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \diamond \mathsf{group}\ (\mathsf{nest}\ 1\ \mathit{symbol\text{-}line})) \\
&\qquad (\mathit{bit\text{-}list}_{\mathsf{P}}\ \mathit{bs})
\end{aligned}
$$

$$\mathit{bs} = \mathsf{true} :: \mathsf{false} :: \mathsf{true} :: \mathsf{true} :: \mathsf{false} :: [\,]$$

$$
\begin{aligned}
&\mathit{render}\ 20\ (\mathit{bit\text{-}list}_{\mathsf{P}}\ \mathit{bs}) \equiv \texttt{"1; 0; 1; 1; 0;"} \\
&\mathit{render}\ 10\ (\mathit{bit\text{-}list}_{\mathsf{P}}\ \mathit{bs}) \equiv \texttt{"1; 0; 1;\textbackslash n 1; 0;"} \\
&\mathit{render}\ 6\ \ (\mathit{bit\text{-}list}_{\mathsf{P}}\ \mathit{bs}) \equiv \texttt{"1; 0;\textbackslash n 1; 1;\textbackslash n 0;"}
\end{aligned}
$$

# Another example

$$
\begin{aligned}
&\mathit{bit\text{-}list}_{\mathsf{P}} : (bs : \mathit{List\ Bool}) \to \mathit{Doc\ bit\text{-}list\ bs} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ [\,] = \mathit{nil} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: [\,]) = \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \diamond \mathit{symbol\text{-}nil})\ \mathit{nil} \\
&\mathit{bit\text{-}list}_{\mathsf{P}}\ (b :: bs) = \\
&\quad \mathit{cons}\ (\mathit{bit}_{\mathsf{P}}\ b \diamond \mathsf{group}\ (\mathsf{nest}\ 1\ \mathit{symbol\text{-}line})) \\
&\qquad (\mathit{bit\text{-}list}_{\mathsf{P}}\ bs)
\end{aligned}
$$

$$bs = \mathsf{true} :: \mathsf{false} :: \mathsf{true} :: \mathsf{true} :: \mathsf{false} :: [\,]$$

$bs \in \mathit{bit\text{-}list} \cdot$ `"1; 0; 1; 1; 0;"`

$bs \in \mathit{bit\text{-}list} \cdot$ `"1; 0; 1;\n 1; 0;"`

$bs \in \mathit{bit\text{-}list} \cdot$ `"1; 0;\n 1; 1;\n 0;"`

# More

- Can use much more general grammar formalism (recursively enumerable languages).
- More advanced examples available (operators with precedence, an XML-like language).
- One can prove that the document combinators satisfy certain algebraic properties.

# Conclusions

- Light-weight approach to correct-by-construction pretty-printing.
- Based on classical pretty-printing, but precisely typed.
- Separates grammars and pretty-printers.
- Seems to work well when the pretty-printer follows the grammar's structure.