Preface

This is a book about hacking in ocaml. It's assumed that you already understand the underlying theory. Happy hacking Most parts are filled with code blocks, I will add some comments in the future. Still a book in progress. Don't distribute it.

☺

Acknowledgements

write later

Contents

Todo list

Chapter 1

Tool Chain

Chapter 2

Lexing

Chapter 3

Parsing

Chapter 4

Camlp4

Camlp4 stands for Preprocess-Pretty-Printer for `OCaml`, it's extremely powerful and hard to grasp as well. It is a source-to-source level translation tool.

Chapter 5

Libraries

5.2 Format

Format is a pretty printer library.

```
open_box 0; print_string "x ="; print_space ();
print_int 1; close_box (); print_newline ()
(** equivalent *)

printf "@[x =@ %i@]@." 1.
```

In case of interactive use, the system closes all opened boxes and flushes all pending text (as with the print_newline function) after each phrase. Each phrase is therefore executed in the initial state of the pretty-printer.

The material output by the following functions is delayed in the pretty-printer queue in order to compute the proper line breaking. Hence, you *should not mix* calls to the printing functions of the basic I/O system with calls to the functions of this module: this could result in some strange output seemingly unrelated with the evaluation order of printing commands.

Some important APIs:

open_box d opens a new pretty-printing box with *offset d*. This box is the general purpose pretty-printing box. Material in this box is displayed *horizontal or vertical*; break hints inside the box may lead to a new line, if there is no more room on the line to print the remainder of the box, or if a new line may lead to a new indentation (demonstrating the indentation of the box). When a new line is printed in the box, *d is added* to the current indentation.

print_as len str prints str in the current box. The *pretty-printer* formats str as if it were of length len

print_space () is used to separate items (typically to print a space between two words). It indicates that the line may be split at this point. It either prints one space or splits the line. It is equivalent to *print_break 1 0*.

print_cut () is used to mark a good break position. It indicates that the line may be split at this point. It either prints nothing or splits the line. This allows line splitting at the current point, without printing spaces or adding indentation. It is equivalent to *print_break 0 0*.

Inserts a break hint in a pretty-printing box. *print_break nspaces offset* indicates that the line may be split (a newline character is printed) at this point, if the contents of the current box does not fit on the current line. If the line is split at that point, offset is added to the current indentation. If the line is not split, nspaces spaces are printed.

Boxes, Break hints, and Indentation rules.

5.2.1 Indentation Rules

A box can state *the extra indentation of every new line* opened in its scope. This extra indentation is named box breaking indentation.

A break hint can also set the additional indentation of the new line it may fire. This extra indentation is named *hint breaking indentation*.

If break hint bh fires a new line within box b, then the indentation of the new line is simply the sum of: the current indentation of box b + the additional box breaking indentation, as defined by box b + the additional hint breaking indentation, as defined by break hint bh.

5.2.2 Boxes

h box, within this box, break hints do not lead to line breaks.

v box, within this box, every break hint leads to a new line.

hv box, if it is possible, the entire box is written on a single line; otherwise, every break hint within the box leads to a new line.

hov box. within this box, break hints are used to cut the line when there is no more room on the line.

Suppose we can write 10 chars before the right margin (that indicates no more room). We represent any char as a - sign; characters [and] indicates the opening and closing of a box and b stands for a break hint given to the pretty-printing engine.

The output "–b–b–" is displayed like this (the b symbol stands for the value of the break that is explained below):

within a "h" box: –b–b–

within a “v” box:
–b
–b
–

within a “hv” box: If there is enough room to print the box on the line: –b–b–
But “—b—b—” that cannot fit on the line is written
—b
—b
—

within a “hov” box: If there is enough room to print the box on the line: –b–b–
But if “—b—b—” cannot fit on the line, it is written as
—b—b
—

Break hints are also used to output spaces. You output a break hint using *print_break* sp indent, and this *sp* integer is used to print “sp” spaces. Thus *print_break sp ...* may be thought as: *print sp spaces or output a new line.*

For instance, if b is break 1 0 in the output “–b–b–”, we get within a “h” box: – – – within a “v” box: –
–
–
within a “hv” box: – – –
or, according to the remaining room on the line: –
–
–
and similarly for “hov” boxes.

5.2.3 Directives

“ “ outputs a breakable space (*print_space ()*)

“,” output a break hiint (*print_cut ()*)

“@;<n m>” emit a “full” break hint (*print_break n m*)

“@.” end the pretty-printing, closing all the boxes still opened (*print_newline ()*)

“@[<hov n>” is equivalent to *open_hovbox n*.

5.2.4 Example: Print

Listing 53: Format

5.16 Modules

Chapter 6

Runtime

Chapter 7

GC

Should be re-written later

Chapter 8

Object-oriented

Write later

Chapter 9

Language Features

9.9 The module Language

write later

Chapter 10

subtle bugs

Chapter 11

Interoperating With C

Write later

Chapter 12

Pearls

Chapter 13

XX

Chapter 14

Topics