Interface for module Prtree

1. This module provides a generic ASCII pretty-printing function for trees, in a way similar to what the Unix command `pstree` does:

```
bash-+-emacs-+-emacsserver
     |       `-ispell
     |-pstree
     `-xdvi.bin
```

2. A tree structure is given as an abstract type t together with a decomposition function *decomp* returning the label of the node and the list of the children trees. Leaves are nodes with no child (i.e. an empty list).

```
module type Tree = sig
  type t
  val decomp : t → string × t list
end
```

3. The functor *Make* takes a tree structure T as argument and provides a single function *print* : *formatter* → $T.t$ → *unit* to print a tree on a given formatter.

```
module Make : functor (T : Tree) → sig
  val print : Format.formatter → T.t → unit
end
```

Module Prtree

4. Tree structures.

```
module type Tree = sig
  type t
  val decomp : t → string × t list
end
```

5. Pretty-print functor.

```
module Make(T : Tree) = struct
```

```
  open Format
```

print_node prints one node and *print_sons* its children. *pref* is the prefix to output at the beginning of line and *start* is the branching drawing ("+-" the first time, and then "|-").

```
  let print fmt t =
    let rec print_node pref t =
      let (s, sons) = T.decomp t in
      pp_print_string fmt s;
      if sons ≠ [] then
        let w = String.length s in
        let pref′ = pref ^ String.make (w + 1) ' ' in
        match sons with
          | [t′] → pp_print_string fmt "---"; print_node (pref′ ^ "␣␣") t′
          | _ → pp_print_string fmt "-"; print_sons pref′ "+-" sons
    and print_sons pref start = function
      | [] →
          assert false
      | [s] →
          pp_print_string fmt "`-"; print_node (pref ^ "␣␣") s
      | s :: sons →
          pp_print_string fmt start; print_node (pref ^ "|␣") s;
          pp_force_newline fmt (); pp_print_string fmt pref;
          print_sons pref "|-" sons
    in
    print_node "" t
end
```