Recent Development Directions for makeinfo

Patrice Dumas

pertusus@gnu.org

GHM 2011

Texinfo and the GNU project

- Texinfo is the "official" documentation format for the GNU project
- Syntax similar with TEX, with structural markup, for technical manuals
- competition:

Docbook	benefits of XML, but lots of markup
LATEX	presentation markup
Asciidoc	less structured

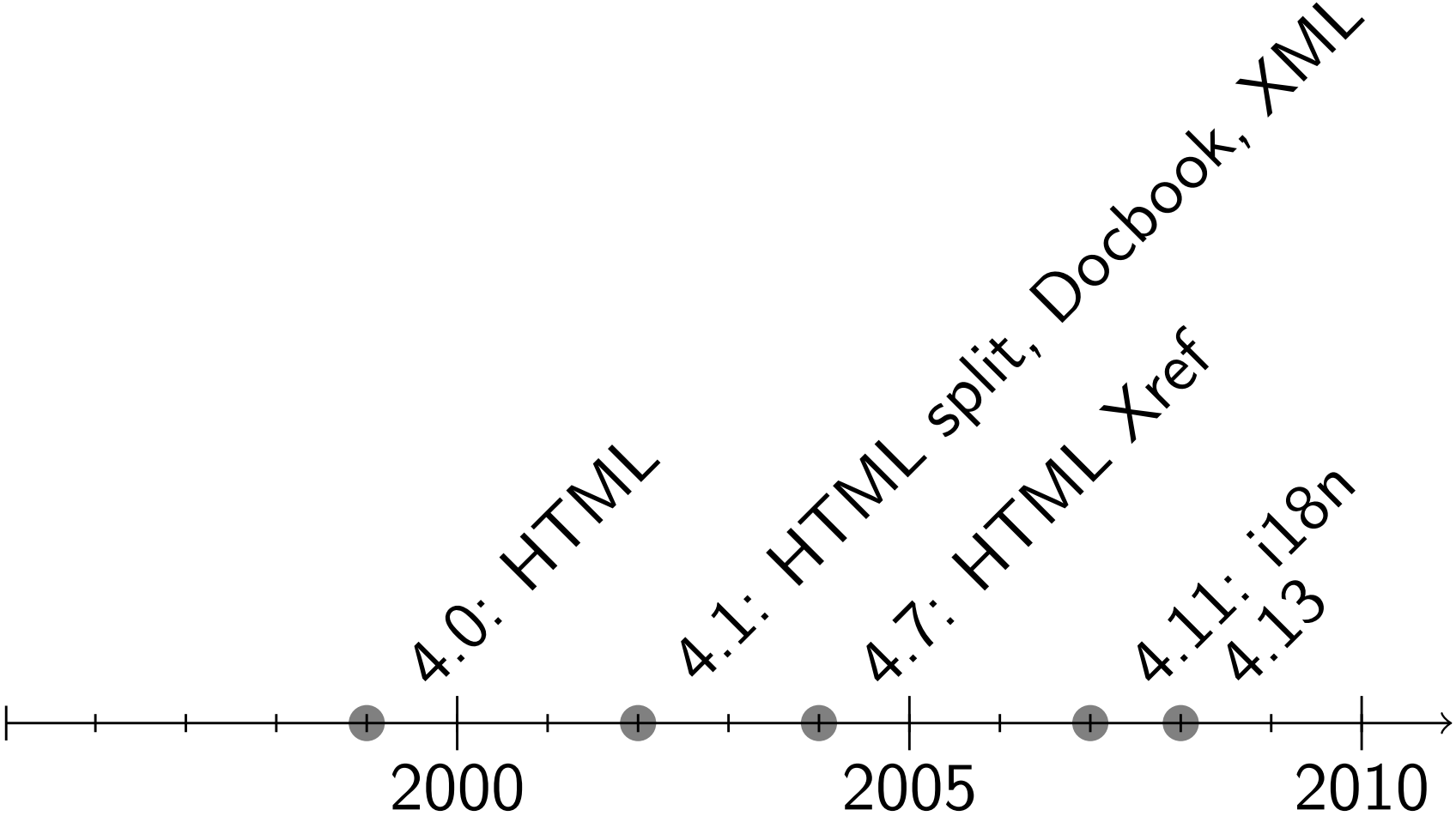
- two converters:

  texi2dvi, texi2pdf plain TEX to convert to dvi, ps, pdf

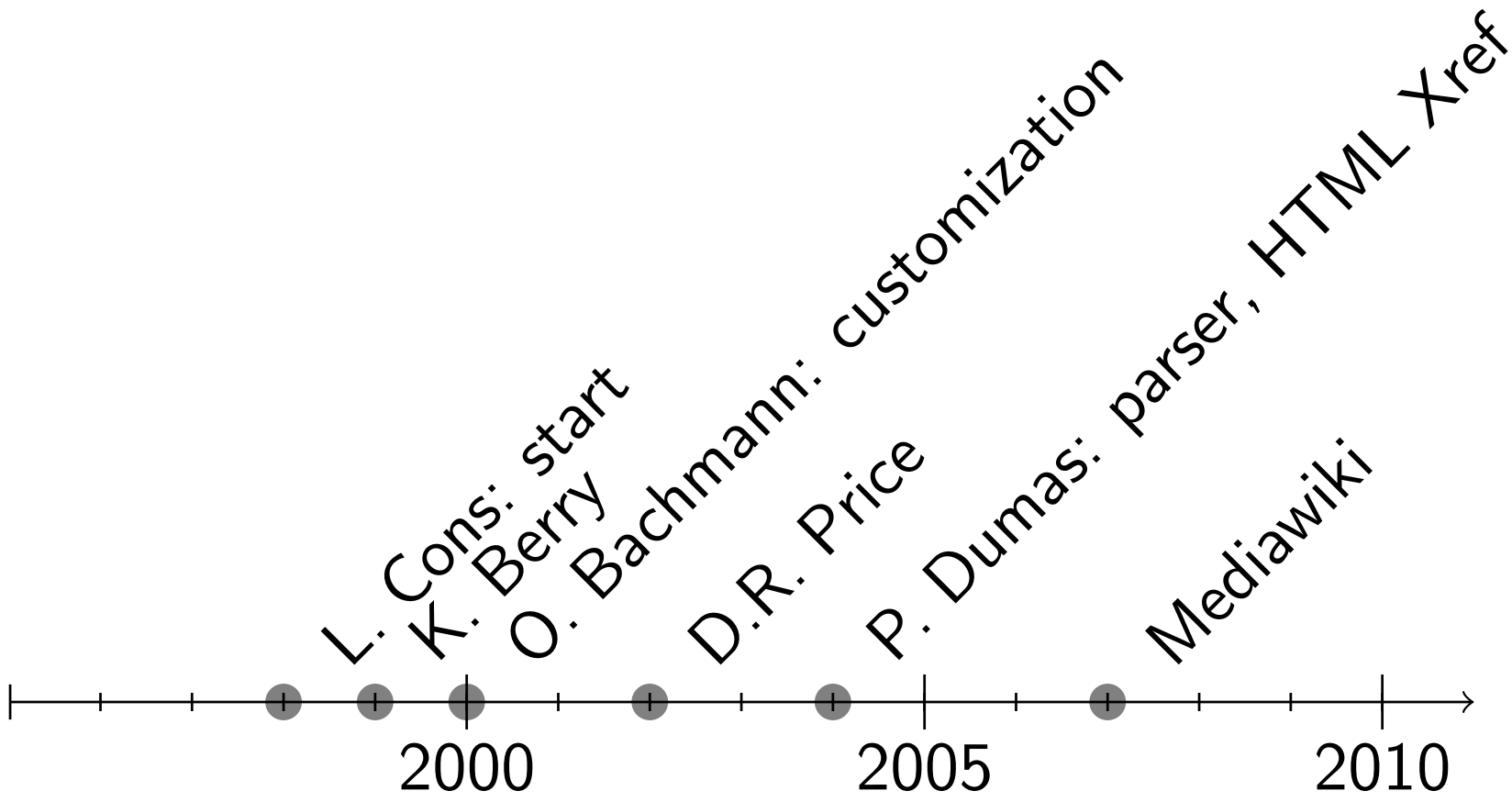
  makeinfo, texi2html, texi2any converts to Info, HTML, Docbook, Texinfo XML

Table of Contents

“Recent” makeinfo development until 2008



In parallel with texi2html



texi2html in texinfo

- lilypond manual switches to texi2html
- July 2008 decision to switch to texi2html in texinfo if made compatible with makeinfo
- September 2008 last texinfo release with C makeinfo installed
- June 2010 last stand alone texi2html release, in texinfo source tree
- November 2010 possibility of a first release with texi2html as makeinfo

Selling points: customization (design)
Problematic points: speed and design.

The new Texinfo Parser development

- September to November 2010 a Texinfo Parser is developped
- November 2010 decision to use the Parser in next release!
- March 2011 enhanced Info backend done
- August 2011 HTML backend done

Selling points: correctness, customization and design.
Problematic point: speed.

Table of Contents

The Parser and backends

- The Parser parses the Texinfo code into a tree
- Additional information is gathered on the tree, like sectioning and nodes trees
- Backends convert the tree to an output format

- Perl modules and a command (makeinfo/texi2any)
- First level of customization through "customization variables". Example USE_ACCESSKEY
- Convert::HTML fully customizable with hashes and functions

The Texinfo tree

Common keys for tree element

- cmdname — the @-command name if corresponding to a command
- type — element type, in general not present with cmdname
- text — a text fragment
- args — array of arguments in @-command brace, or line, or menu entry
- contents — array, all purpose container, holding paragraph content, argument content, block command content
- parent — the parent element
- extra — placeholder for any other information

Other keys with line numbers, and directions for sectioning commands and nodes.

Example of tree

```
@quotation
Text @@ @code{c}
@end quotation
```

```
{'cmdname' => 'quotation',
 'contents' => [
   {'type' => 'empty_line_after_command'}
    'text' => '
'},
   {'type' => 'paragraph',
    'contents' => [
      {'text' => 'Text␣'},
      {'cmdname' => '@'},
      {'text' => '␣'},
      { 'cmdname' => 'code',
        'args' => [
         {'type' => 'brace_command_arg'
          'contents' => [ {'text' => 'c'} ] } ] },
      {'text' => '
'}]}]}
```

HTML customization

Functions references called for elements. Defined in customization files.

```
$commands_conversion{'quotation'} = \&my_quotation;

sub my_quotation($$$$$)
{
  my $converter = shift; # opaque object
  my $cmdname = shift;   # the @-command cmdname
  my $command = shift;   # the tree object
  my $args = shift;      # formated arguments array
  my $content = shift;   # formatted content

  my $result = '';
  if ($command->{'extra'}
      and $command->{'extra'}->{'authors'}) {
    $result = 'With author';
  }
  $result .= 'pre ' if ($converter->in_preformatted());
  $result .= $args->[0]->{'normal'} if ($args->[0]);
  return $result .$content;
}
```

The future

- a release as soon as possible (docbook, Texinfo XML)

- @node and @menu automatic generation
- re-enable math formatting through latex2html/tex4ht
- maybe revive Mediawiki, roff outputs
- finish LaTeX backend
- epub, odp?

- use m4 to preprocess the Texinfo
- more simple math commands, and math environments