Literate Proofs

Ruben Gamboa

July 14, 2003

Motivation

- Make it easier to understand ACL2 books after the fact
- Organize ACL2 books in a human-oriented, not a proof-oriented manner
- Publish ACL2 books in different formats (web, print, ...) and different styles (ACL2 workshop, Nqthm, ...)

Literate Programming

- We follow literate programming paradigm:
  - Source code and documentation in the same source document
  - Tools extract the source code and documentation for further processing
- Originally invented by Knuth to make Pascal programming less painful

Literate Programming Tools

- Two tools are needed
  - Extract source code
  - Extract documentation
- Knuth wrote the programs “tangle” and “weave” for this purpose

Using XML Tools

- Tangle and Weave
  - process a text file
  - identify chunks from the text file
  - possibly reorder the chunks
  - possibly reformat the chunks
- XML tools can do all that

XML Transformations

- XSLT is the standard mechanism for specifying XML transformations
- These transformations can do everything we need for literate programming
- Our work is based on Norm Walsh's XSLT stylesheets for literate programming

A Sample File

```
<article>
  <para>
    We begin by defining append.
  </para>

  <src:fragment id="top">
    (defun my-append (x y)
      (if (endp x)
          y
        (cons (car x) (my-append (cdr x) y))))
    <src:fragref linkend="proofs">
  </src:fragment>
  ...
</article>
```

Extracting the ACL2 Proof Script

- The XSLT stylesheets extract the <src:fragment> chunk named “top”
- Chunks referenced with <src:fragref> are inserted into the output
- Order of ACL2 proof script is not necessarily the same as the order in the original document

Extracting the Documentation

- Notice that we have two types of <tag>s
  - <src:...> tags
  - The remaining tags
- The remaining tags can be in any XML dialect, e.g., HTML, DocBook, ...
- We need to convert <src:...> tags to the same XML dialect as the remaining tags

Extracting into HTML

- <src:fragment> can be converted to HTML <pre> tags
- <src:fragref> can be converted into hyperlinks to the appropriate section

Extending the <src:...> Markup

- So far, our tools know nothing of the ACL2 code inside <src:fragment>s
- By adding more <src:...> markup, we can customize our tools to provide different presentations of the ACL2 code

Another Example

```
<src:defun function="my-append">
  <src:arg>x</src:arg>
  <src:arg>y</src:arg>
  <src:body>
    (if (endp x)
        y
      (cons (car x)
            (my-append (cdr x) y)))
  </src:body>
</src:defun>
```

Different Output Styles

- Now our stylesheets know when they are processing a definition
- They can render this definition in different ways, e.g.
  - Traditional ACL2 syntax
  - Nqthm-style syntax

More <src:...> Tags

- We can add tags for any ACL2 feature we may want to special-case
  - Hints
  - Rule-Classes
  - Documentation strings
- We must specify how to convert each new tag into ACL2 and the surrounding XML

Examples

- The best way to look at this work is to browse through some examples
  - http://www.cs.uwyo.edu/~ruben/projects/litproofs