mathaction2input

Bill Page and David Cygański

5 July 2007

Abstract

This script converts the source text of a mathaction (Axiom Wiki) page containing text, Axiom interpreter commands, SPAD, and Aldor library code.

Contents

1 Makefile

⟨*⟩≡
```
#!/usr/bin/perl
#
# Name: mathaction2input
# Version: 0.2
# Authors: Bill Page and David Cyganski
# Date: 5 July 2007
#
# Description:
# This script converts the source text of a mathaction (Axiom Wiki) page
# containing text, Axiom interpreter commands, SPAD, and Aldor library
# code.
#
# Example:
# $ curl http://wiki.axiom-developer.org/SandBoxTest2/src | \
#        ./mathaction2input > test2.input
#
# ---page: SandBoxTest2 --------
# This is an example:
# \begin{axiom}
# integrate(sin(x),x)
# \end{axiom}
#
# We can also include SPAD
# \begin{spad}
# )abbrev package MYPACK MyPackage
# MyPackage(): with ...
# \end{spad}
#
# and Aldor routines
# \begin{aldor}[name]
# #include "axiom"
# ...
# ---end -------------------------------
#
# This will be converted to a form suitable for direct input to Axiom using
# the ')read test2.input' command. E.g.
#
# ---file: test2.input -------------------
# --This is an example:
# integrate(sin(x),x)
#
# --We can also include SPAD
# )compile MyPackage.spad
```

```
#
# --and Aldor routines
# )compile name.as
# ---end ------------------------------
#
# ---file: MyPackage.spad --------------
# )abbrev package MYPACK MyPackage
# MyPackage(): with ...
# ---end -------------------------------
# ---file: name.as ----------------------
# #include "axiom"
# ...
# ---end -------------------------------
#
# Related: input2mathaction
#

$axiomenv="";  # state variable indicates text/axiom/spad/aldor environment
$filename="";  # name for spad and aldor compiler files
srand(); $filenum=int(rand(99999999)); # used of anonymous aldor modules
while (<>) {
if ($axiomenv eq "") { # currently in text comments
if (m/^[\t ]*\\begin\{axiom\}/) { # start axiom commands
$axiomenv="axiom"
} elsif (m/^[\t ]*\\begin\{spad\}/) { # start spad code
$axiomenv="spad"
} elsif (m/^[\t ]*\\begin\{aldor\}[\t ]*\n/) { # start aldor code
$axiomenv="aldor";                          # anonymous module
while (-f "aldor-$filenum.as") { $filenum=int(rand(99999999)) };
$filename="aldor-$filenum.as";
open(F,">$filename") or die "Can't write to: $filename\n";
print ")compile $filename\n";
} elsif (m/^[\t ]*\\begin\{aldor\}[\t ]*\[(.*?)\][\t ]*\n/) {
$axiomenv="aldor";                          # named aldor module
$filename="$1.as";
open(F,">$filename") or die "Can't write to: $filename\n";
print ")compile $filename\n";
} elsif (m/^[\t ]*\\end\{/) { #  does not belong here!
die ("Nested $_")
} else {
if ($_ ne "\n") {   # text become comments in input file
print "--$_"
} else {            # except blank lines
print
}
}
```

```
} elsif ($axiomenv eq "axiom") {  # current in axiom commands
if (m/^[\t ]*\\begin\{/) {
die ("nested $_")
} elsif (m/^[\t ]*\\end\{axiom\}/) {
$axiomenv=""        # assume next mode is text
} else {
print               # axiom commands go to stdout
}
} elsif ($axiomenv eq "spad") {  # currently in spad code section
if (m/^[\t ]*\\begin\{/) {
die ("\nNested $_")
} elsif (m/^[\t ]*\)abbrev (.*?) (.*?) (.*?)\n/) {
$filename = "$3.spad";
open(F,">$filename") or die "Can't write to: $filename\n";
print F $_;
print ")compile $filename\n";
} elsif (m/^[\t ]*\\end\{spad\}/) {
$axiomenv="";           # assume next mode is text
close(F);
$filename="";
} else {
if ($filename ne "") {  # code goes to a file
print F $_
} else {
die "Missing )abbrev ...\n"
}
}
} elsif ($axiomenv eq "aldor") {  # currently in aldor code section
if (m/^[\t ]*\\begin\{/) {
die ("\nNested $_")
} elsif (m/^[\t ]*\\end\{aldor\}/) {
$axiomenv="";           # assume next mode is text
close(F);
$filename="";
} else {
if ($filename ne "") {  # code goes to a file
print F $_
} else {
die "Program error: No aldor file name\n"
}
}
} else {
die ("\nProgram error: No $axiomenv\n")
}
};
die ("\nMissing \\end{$axiomenv}\n") if $axiomenv;
```

References

[1] nothing