Formal Specification of Abstract Datatypes
Exercise 1 (April 7)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

The result is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file which contains

- a PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - a section with the source code of the specifications,
  - a section with at least six specification tests (sample reductions),
  - optionally any explanations or comments you would like to make;
- the CafeOBJ (.mod) file(s) of the specifications.

Exercise 1: A Simple File System

Write a CafeOBJ specification for a file system consisting of sorts *Path*, *File*, and *FileSystem*.

A *Path* represents the location of a file by operations

$$\begin{aligned}
&\mathit{root} : \mathit{Path}\\
&\mathit{path} : \mathit{Path} \times \mathit{String} \to \mathit{Path}\\
&\mathit{dirpath} : \mathit{Path} \to \mathit{Path}\\
&\mathit{basename} : \mathit{Path} \to \mathit{String}\\
&\ldots
\end{aligned}$$

where e.g. path "/tmp/t.txt" is represented as $\mathit{path}(\mathit{path}(\mathit{root}, \text{“tmp”}), \text{“t.txt”})$; the directory part of that path is $\mathit{path}(\mathit{root}, \text{“tmp”})$, the base name of that path is "t.txt"; model the operations correspondingly.

A *File* consists of a *Path* (the location of the file) and a *String* (the content of the file); it can be accessed by operations

$\mathit{file} : \mathit{Path} \times \mathit{String} \to \mathit{File}$
$\mathit{name} : \mathit{File} \to \mathit{Path}$
$\mathit{text} : \mathit{File} \to \mathit{String}$

A *FileSystem* maps paths to files

$\mathit{emptyFS} :\to \mathit{FileSystem}$
$\mathit{write} : \mathit{FileSystem} \times \mathit{Path} \times \mathit{String} \to \mathit{FileSystem}$
$\mathit{get} : \mathit{FileSystem} \times \mathit{Path} \to \mathit{File}$
$\mathit{read} : \mathit{FileSystem} \times \mathit{Path} \to \mathit{String}$
$\mathit{remove} : \mathit{FileSystem} \times \mathit{Path} \to \mathit{FileSystem}$

An application $\mathit{write}(\mathit{fs}, p, t)$ updates file system *fs* by creating a file with path p and content t (overwriting any previous file with that path). An application $\mathit{get}(\mathit{fs}, p)$ returns the file with path p in *fs* (if such a file exists); an application $\mathit{read}(\mathit{fs}, p)$ returns the content of this file (if the file exists). The operation $\mathit{remove}(\mathit{fs}, p)$, removes any entity whose path starts with (or is identical to) p from the file system.

Write separate tight CafeOBJ modules `PATH`, `FILE`, and `FILESYSTEM` with the corresponding sorts and operations; whenever necessary, you may introduce auxiliary operations (in addition to those specified above). In the specification of one operation, try to (re)use already specified other operations, as far as possible. As a hint, the only operations that demands a “recursive” specification are *get* and *remove*, all other operations can be specified in a “direct” way.

Test the specifications with several reductions, among them the following two (this is *not* actual CafeOBJ syntax):

let $h = \mathit{path}(\mathit{path}(\mathit{root}, \text{“home”}), \text{“ws”})$
let $u = \mathit{path}(\mathit{root}, \text{“usr”})$
let $p1 = \mathit{path}(h, \text{“t.txt”})$
let $p2 = \mathit{path}(h, \text{”u.txt”})$
let $p3 = \mathit{path}(u, \text{”v.dat”})$
let $fs1 = \mathit{write}(\mathit{emptyFS}, p1, \text{“hello”})$
let $fs2 = \mathit{write}(\mathit{fs1}, p2, \text{“hi”})$
let $fs3 = \mathit{write}(\mathit{fs2}, p1, \text{“greetings”})$
let $fs4 = \mathit{write}(\mathit{fs3}, p3, \text{“xxx”})$
$\mathit{read}(\mathit{fs4}, \mathit{p1})$
$\mathit{read}(\mathit{fs4}, \mathit{p2})$
$\mathit{read}(\mathit{fs4}, \mathit{p3})$
let $fs5 = \mathit{remove}(\mathit{fs4}, u)$
$\mathit{read}(\mathit{fs5}, \mathit{p1})$
$\mathit{read}(\mathit{fs5}, \mathit{p2})$
$\mathit{read}(\mathit{fs5}, \mathit{p3})$

Give the input and output of each test and your interpretation of the output (does it indicate an error in your specification or not?). If your specification contains an error, use the trace facilities of CafeOBJ to detect the source of the error.