

# Formal Specification of Abstract Datatypes

## Exercise 1 (April 19)

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

March 18, 2010

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

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

```
root : Path
path : Path × String → Path
dirpath : Path → Path
basename : Path → String
...
```

where e.g. path “/tmp/t.txt” is represented as  $path(path(root, “tmp”), “t.txt”)$ ; the directory part of that path is  $path(root, “tmp”)$ , 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

```

nullFile :→ File
file : Path × String → File
name : File → Path
text : File → String
...

```

A *FileSystem* maps paths to files

```

emptyFS :→ FileSystem
write : FileSystem × Path × String → FileSystem
get : FileSystem × Path → File
read : FileSystem × Path → String
...

```

An application  $write(fs, p, t)$  updates file system  $fs$  by creating a file with path  $p$  and content  $t$  (overwriting any previous file with that path). An application  $get(fs, p)$  returns the file with path  $p$  in  $fs$  (or the *nullFile*, if no such file exists); an application  $read(fs, p)$  returns the content of this file (if the file exists).

Write 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).

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

```

let h = path(path(root, "home"), "ws")
let p1 = path(h, "t.txt")
let p2 = path(h, "u.txt")
let p3 = path(h, "v.txt")
let fs1 = write(emptyFS, p1, "hello")
let fs2 = write(fs1, p2, "hi")
let fs3 = write(fs2, p1, "greetings")
read(fs3, p1)
read(fs3, p2)
read(fs3, p3)

```