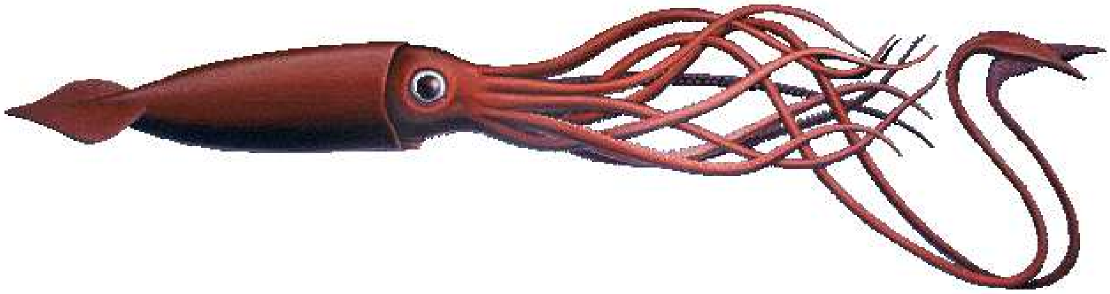
2001 Aldrich Interdisciplinary Lecture and Conference



A Comparison of Two Functional Programming Languages

Donald Craig
Department of Computer Science
February 2001

Programming Techniques

- Imperative (procedural) Languages:
  - Programs are sequence of commands.
  - Uses *assignment* and *procedure*.
  - Flow of control flow is explicit using, for example, `while` and `for` loops.
  - Includes languages such as C, Modula and Pascal.

- Functional (applicative) Languages:
  - Program is a series of function definitions.
  - The fundamental operation is an application of a function.
  - Recursion provides the primary means of control flow.
  - Includes languages such as Haskell, Miranda and ML.

Functional Programming Examples

- A simple function to square its input:

```
square :: Integer -> Integer
square x = x * x

Main> square 5
25
```

- A function to compute factorials:

```
fact :: Integer -> Integer
fact n | n == 0 = 1
       | n < 0  = error "-ve argument"
       | n > 0  = n * fact (n - 1)

Main> fact 5
120
```

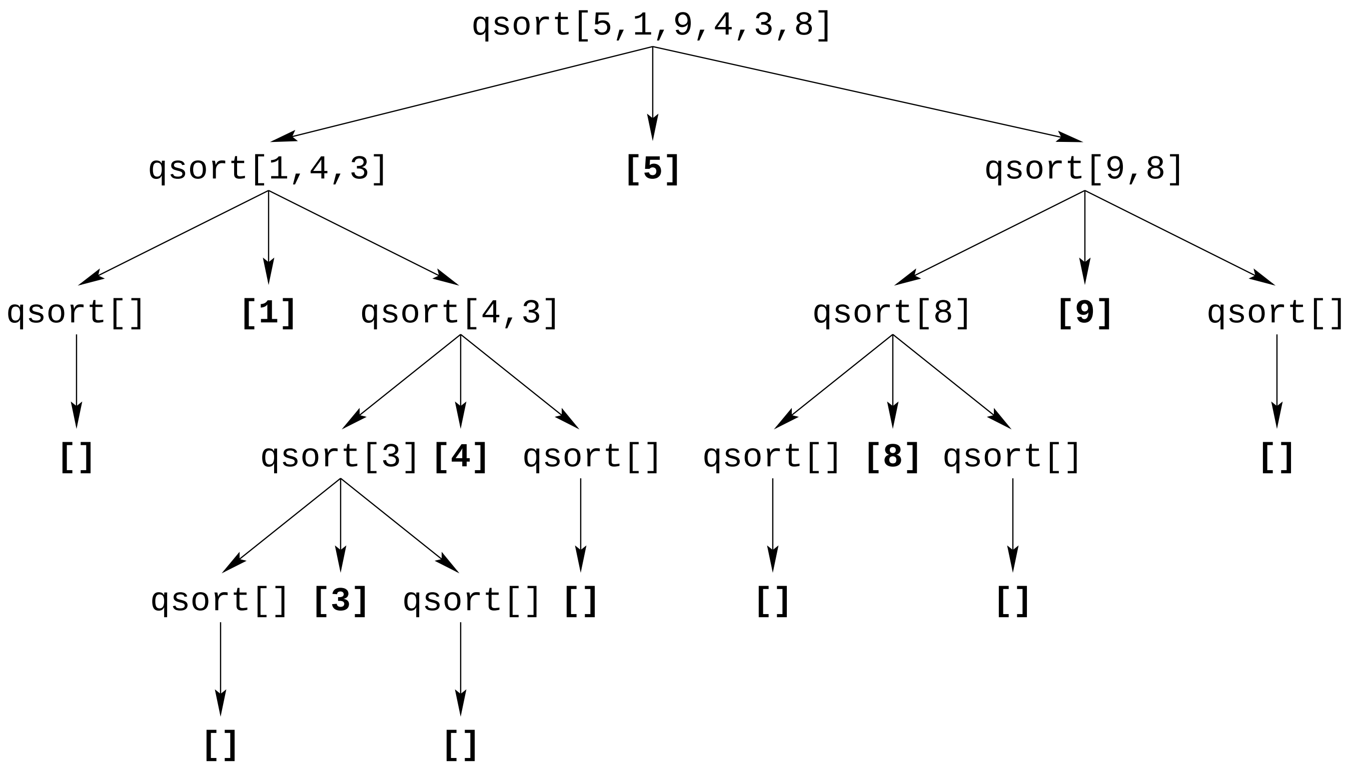
quicksort: C implementation

```
int
partition (int A[], int p, int r)
{
        int         x = A[p];
        int         i = p - 1;
        int         j = r + 1;

        while (1) {
                do -- j; while (A[j] > x);
                do ++ i; while (A[i] < x);
                if (i < j) {
                        int tmp = A[i];
                        A[i] = A[j];
                        A[j] = tmp;
                } else {
                        return j;
                }
        }
}

void
quicksort (int A[], int p, int r)
{
        if (p < r) {
                int         q = partition(A, p, r);
                quicksort (A, p, q);
                quicksort (A, q + 1, r);
        }
}
```

Notice the proliferation of low-level details with respect to *how* the algorithm is implemented.

```
qsort :: Ord a => [a] -> [a]
qsort []       = []
qsort (a:x)    = qsort [b | b <- x, b <= a]
                 ++ [ a ] ++
                 qsort [b | b <- x, b > a]
```

The following tree demonstrates how the `qsort` function partitions the input list and recursively applies the same sorting strategy to each sub-list:



[]++[1]++[]++[3]++[]++[4]++[]++[5]++[]++[8]++[]++[9]++[]

[1,3,4,5,8,9]

Haskell and Miranda

- Syntactic differences.
- Differences in the underlying type systems.
- Relative runtime performance of implementations of Haskell and Miranda.

Syntactic Differences

The influence that Miranda has had upon the development of Haskell can be seen in the similarity of their respective syntaxes. There are, however, some minor differences.

- The guard (conditional) syntax is different:

  ```
  fact n = 1, n = 0
         = error "-ve argument", n < 0
         = n * fact (n - 1), n > 0
  ```

  Haskell also allows the use of `if` ... `then` ... `else` syntax for representing conditionals.

- List comprehensions:

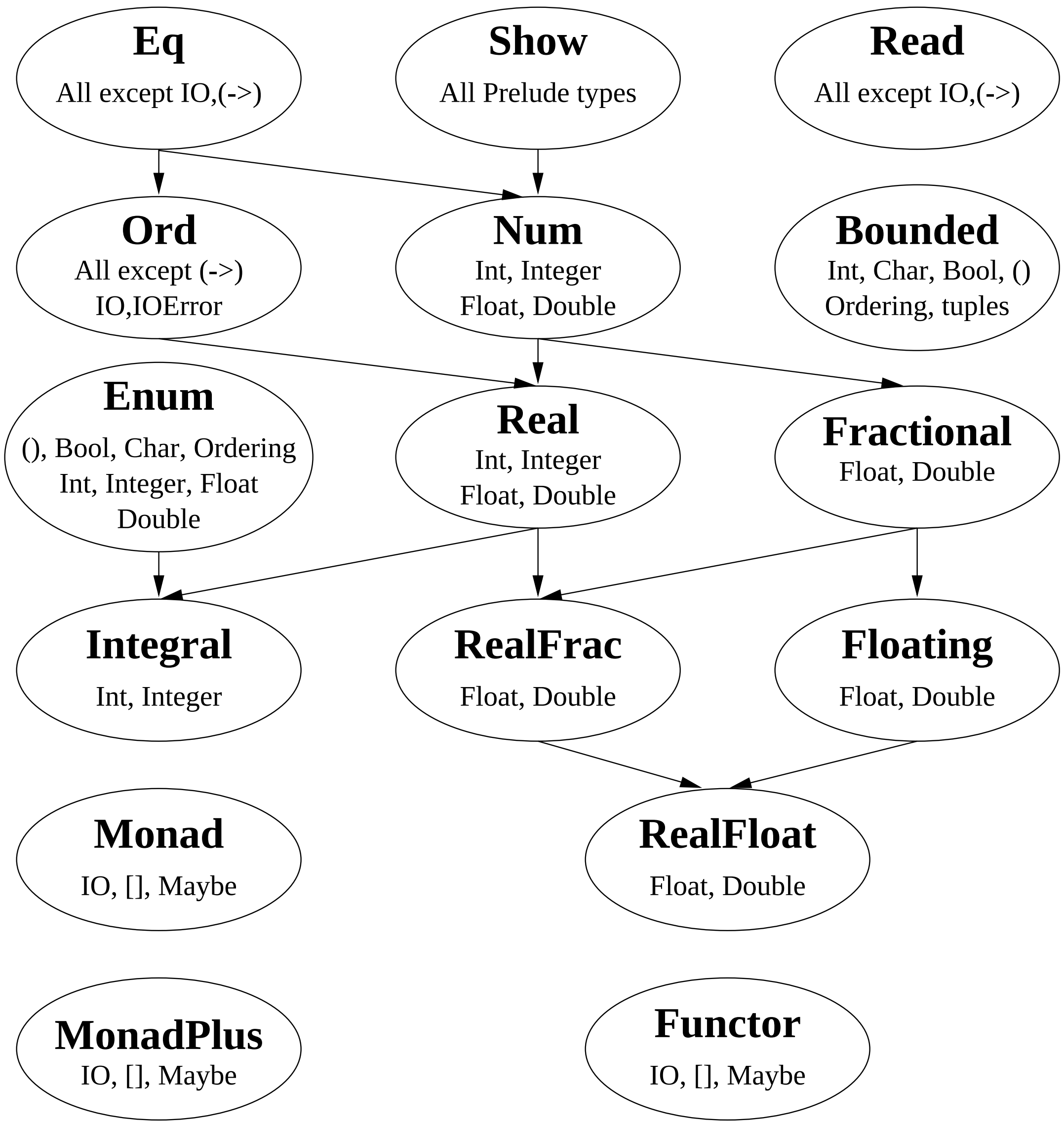
  - Slightly different syntax:

    Miranda: `[ b | b <- x; b <= a ]`
    Haskell: `[ b | b <- x, b <= a ]`

  - Miranda supports recurrence relations; Haskell does not:
    `fibs = [ a | (a,b) <- (1,1), (b,a+b) .. ]`

Haskell Type Classes

Haskell supports the concept of a “type class” which gives one finer control over the types employed by a Haskell script. Type classes can be confusing, however, to one accustomed to a “flat” type hierarchy.



Runtime Performance

The following tables provide some crude empirical results comparing `hugs98` (Haskell Users Gofer System) February 2000 release with Miranda version 2.025.

The results were obtained by having both the respective Haskell and Miranda implementations quicksort 10,000 integers.

Language	Compile Time (sec)	Eval. Time (sec)	Total Time (sec)
Miranda	–	2.82	2.82
Haskell	39.36	9.21	48.57

Language	Reductions	Reductions per sec	Cells Used
Miranda	1,294,200	458,936	1,451,192
Haskell	3,015,776	327,588	3,981,287

Conclusions

- Functional programming languages allow for efficient algorithmic expression in contrast to more popular, imperative languages.

- Proving correctness is much easier for functional programs than imperative ones.

- Despite the expressive efficiency of functional programming, functional programming faces many challenges in the pragmatic world (e.g. runtime efficiency).

- Haskell and Miranda are similar:

  - The minor syntactic differences are not a big problem.

  - The differences in the type systems take more time to digest.

  - The Haskell implementation studied has some performance issues (but a new version of `hugs98` was recently released).

Technical Reports

- *A Comparison of the Haskell and Miranda Functional Programming Languages*

  Technical Report 2000–02

- *Compiling and Using the* `hugs98` *Interpreter*

  Technical Report 2000–03