CSC330 Summer 2005

Introduction to Functional Programming

Functional Languages

- Black box view
- Function y=f(x) f: X->Y
  - Domain X, range Y
  - x independent variable, y dependent variable
- Function defition, application

Functional Languages

- Scheme, ML, Haskell
- AI, prototyping, proof-systems
- Advantages
  - Uniform view of programs as functions
  - Automatic memory management
  - Great flexibility, conciseness of notation and simple semantics
  - Drawback (used to be)
    - Performance

Then why don't I know about them

- Persistance of established technology
- More abstract and mathematical
- Object-oriented programming mirros everyday experience and therefore can be simpler for certain problems
- Less libraries although rapidly catching up

Side Effects – The enemy

- In pure functional languages there are no assignmnets only bindings
- Referential transparency
  - The value of a function depends only on the function and it's arguments
- Value semantics
- Functions are first class citizens

Why SML ?

- Very powerful in expressing structured data
- Simple interactive interface
- Higher-order functions
- Polymorphism
- Freedom from side-effects
- Strong typing with type inference
- Abstract data types

Why SML II ?

- Recursion
- Rule-based programming
- Complete, formal semantics
- Combination of the best (IMHO) features of
  - Lisp
  - Pascal, Modula-2
  - Prolog
  - C++, Smalltalk

FP in the real world

- Phone calls in the European Parliament
  - Erlang (Ericcson's functional language)
- CDs shipped by Polygram in Europe using Software AGs Natural Expert
- FFTW (Fast Fourier Transform in the West)
- HOL Theorem prover
  - Helped design the HP 9000 line of multiprocessors