CSE 341: Programming Languages

Dan Grossman

Spring 2008

Lecture 12— Parametric Polymorphism; Equivalence

Today

Two more "conceptual" topics

- Higher density of more abstract concepts as course progresses
- Think about the theory and how languages "fit together", not just how do I "code something up"

1. Parametric polymorphism
   - Also: Type constructors (e.g., ML's `list` and `option`)
2. Equivalence
   - When are two functions or other expressions "the same"

Parametric Polymorphism

Fancy phrase for "forall types" or sometimes "generics." In ML since mid-80s and now in Java, C#, VB, etc.

- (C++ templates are more like macros (later)).

In ML, there's an implicit "for all" at the beginning of any type with `'a`, `'b`, etc. Example:

```
('a * 'b) -> ('b * 'a)
```

really means:

```
forall 'a, 'b . (('a * 'b) -> ('b * 'a))
```

(though `forall` is just for lecture purposes; it is not in ML)

We can *instantiate* the *type variables* to get a *less general* type. For example, with `string` for `'a` and `int->int` for `'b` we get:

```
(string * (int -> int)) -> ((int->int) * string)
```