New topic: Type inference

What type inference accomplishes

```
-> (define     double (x)         (+ x x))
double                          ;; uScheme
-> (define int double ([x : int]) (+ x x))
double : (int -> int)           ;; Typed uSch.
-> (define     double (x)         (+ x x))
double : (int -> int)           ;; nML
```

What else type inference accomplishes

```
-> ([@ cons bool] #t ([@ cons bool] #f [@ '() bool]))
(#t #f) : (list bool)     ;; typed uScheme
-> (   cons        #t (   cons        #f    '()       ))
(#t #f) : (list bool)     ;; nML
```

How it works

1. For each unknown type, a fresh type variable
2. Every typing rule adds equality constraints
3. Instantiate every variable automatically
4. Introduce polymorphism at `let/val` bindings

Examples

At the board. . .