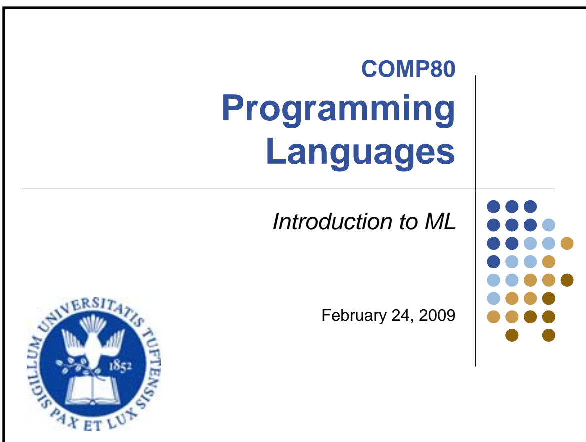
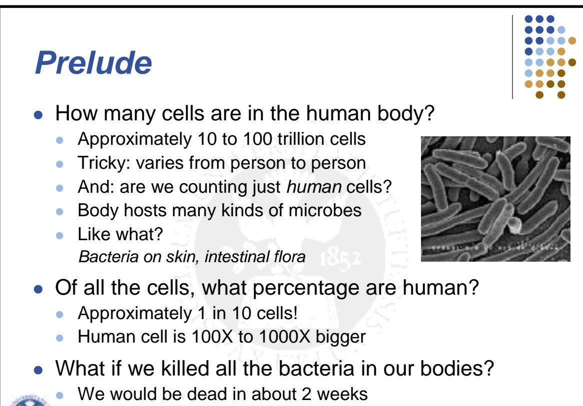
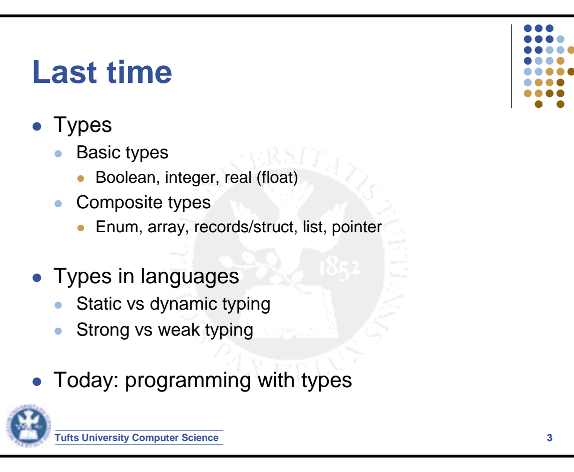
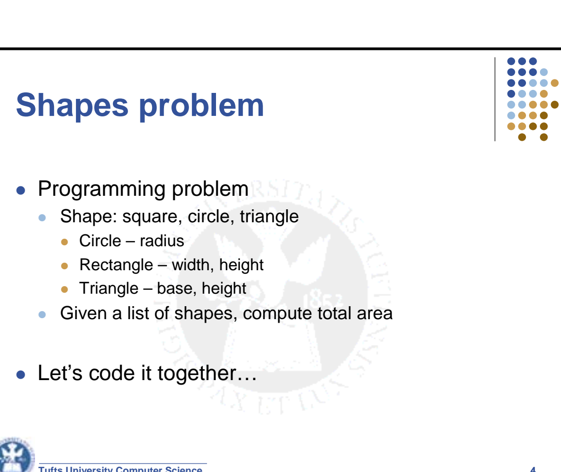
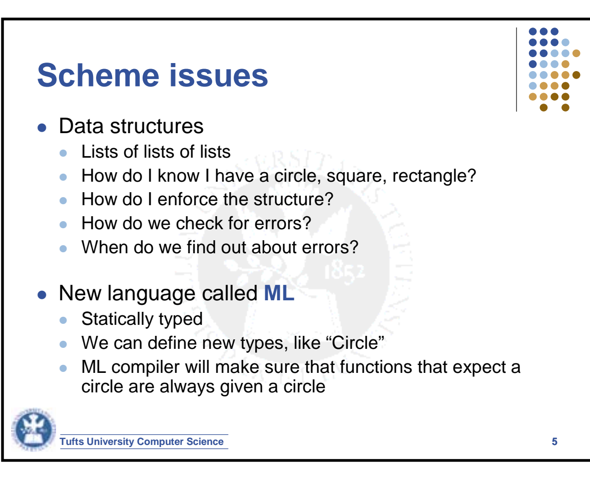
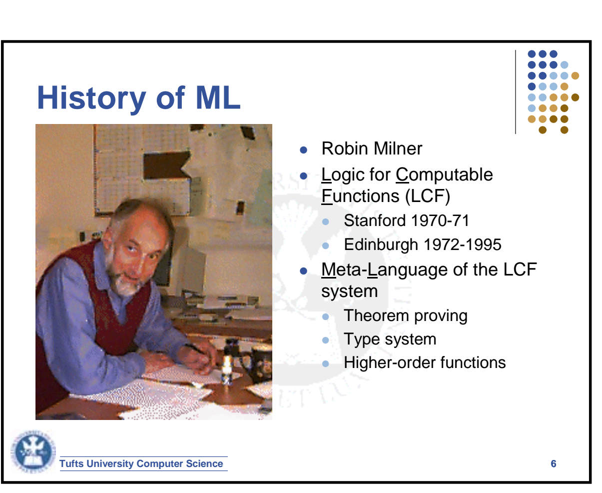

COMP80
Programming Languages
Introduction to ML
February 24, 2009
SIGILLUM UNIVERSITATIS TUFTENSIS
PAX ET LUX
1852

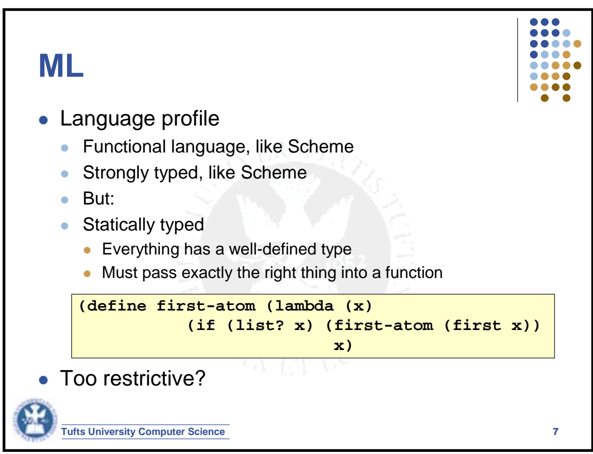

Prelude
How many cells are in the human body?
Approximately 10 to 100 trillion cells
Tricky: varies from person to person
And: are we counting just human cells?
Body hosts many kinds of microbes
Like what?
Bacteria on skin, intestinal flora
Of all the cells, what percentage are human?
Approximately 1 in 10 cells!
Human cell is 100X to 1000X bigger
What if we killed all the bacteria in our bodies?
We would be dead in about 2 weeks
Tufts University Computer Science
2


Last time
Types
Basic types
Boolean, integer, real (float)
Composite types
Enum, array, records/struct, list, pointer
Types in languages
Static vs dynamic typing
Strong vs weak typing
Today: programming with types
Tufts University Computer Science
3

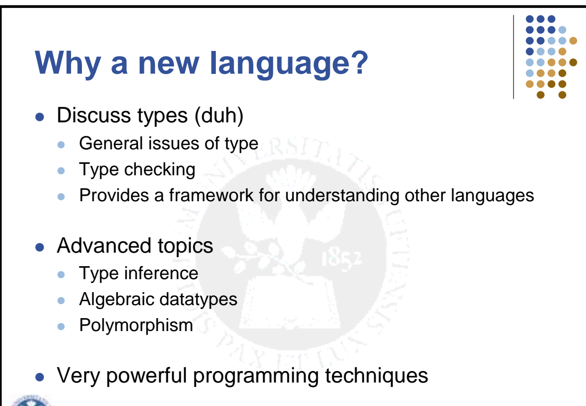

Shapes problem
Programming problem
Shape: square, circle, triangle
Circle – radius
Rectangle – width, height
Triangle – base, height
Given a list of shapes, compute total area
Let's code it together…
Tufts University Computer Science
4


Scheme issues
Data structures
Lists of lists of lists
How do I know I have a circle, square, rectangle?
How do I enforce the structure?
How do we check for errors?
When do we find out about errors?
New language called ML
Statically typed
We can define new types, like "Circle"
ML compiler will make sure that functions that expect a circle are always given a circle
Tufts University Computer Science
5

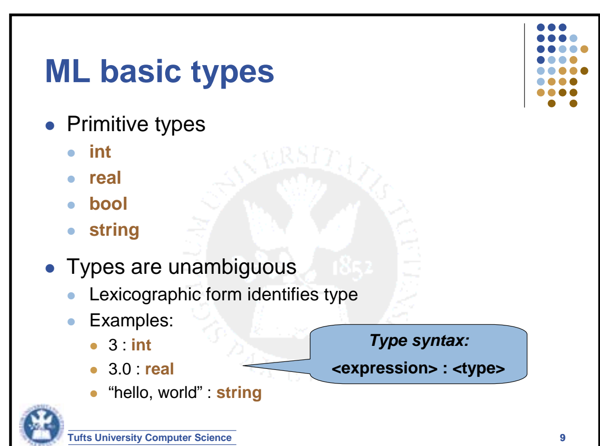

History of ML
Robin Milner
Logic for Computable Functions (LCF)
Stanford 1970-71
Edinburgh 1972-1995
Meta-Language of the LCF system
Theorem proving
Type system
Higher-order functions
Tufts University Computer Science
6



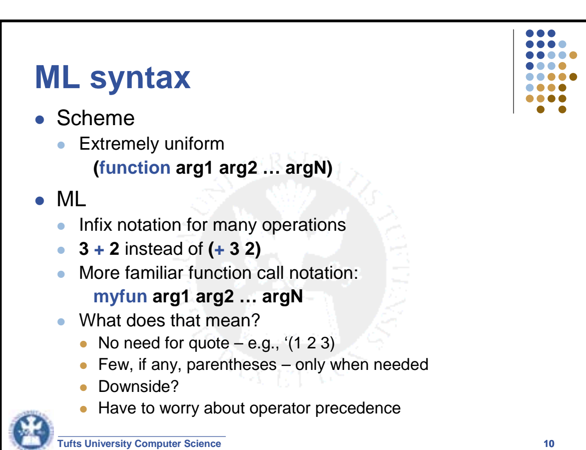
ML
Language profile
Functional language, like Scheme
Strongly typed, like Scheme
But:
Statically typed
Everything has a well-defined type
Must pass exactly the right thing into a function
(define first-atom (lambda (x)
(if (list? x) (first-atom (first x))
x)
Too restrictive?
Tufts University Computer Science
7



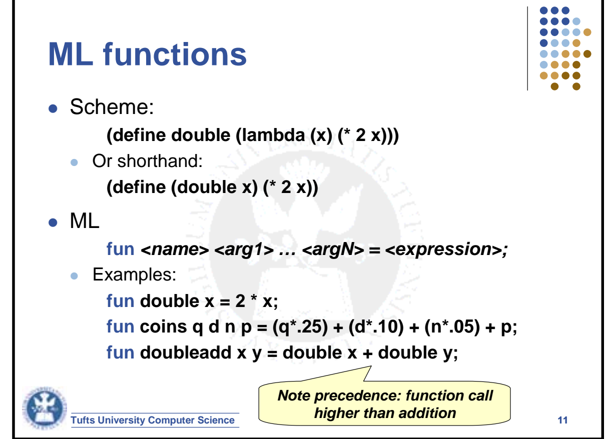
Why a new language?
Discuss types (duh)
General issues of type
Type checking
Provides a framework for understanding other languages
Advanced topics
Type inference
Algebraic datatypes
Polymorphism
Very powerful programming techniques
Tufts University Computer Science
8



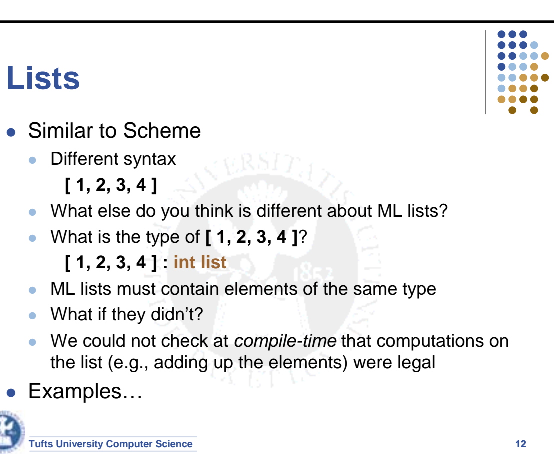
ML basic types
Primitive types
int
real
bool
string
Types are unambiguous
Lexicographic form identifies type
Examples:
3 : int
3.0 : real
"hello, world" : string
Type syntax:
<expression> : <type>
Tufts University Computer Science
9



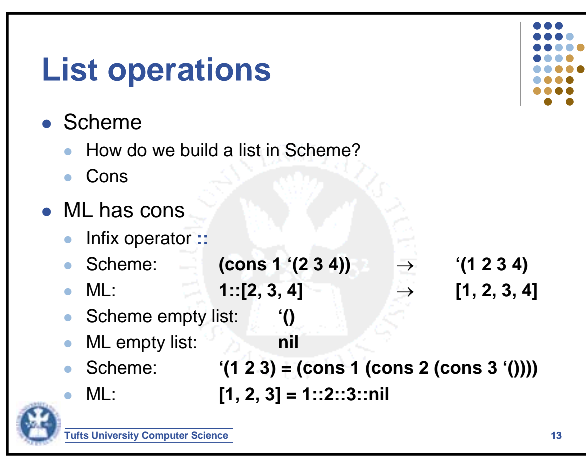
ML syntax
Scheme
Extremely uniform
(function arg1 arg2 … argN)
ML
Infix notation for many operations
3 + 2 instead of (+ 3 2)
More familiar function call notation:
myfun arg1 arg2 … argN
What does that mean?
No need for quote – e.g., '(1 2 3)
Few, if any, parentheses – only when needed
Downside?
Have to worry about operator precedence
Tufts University Computer Science
10

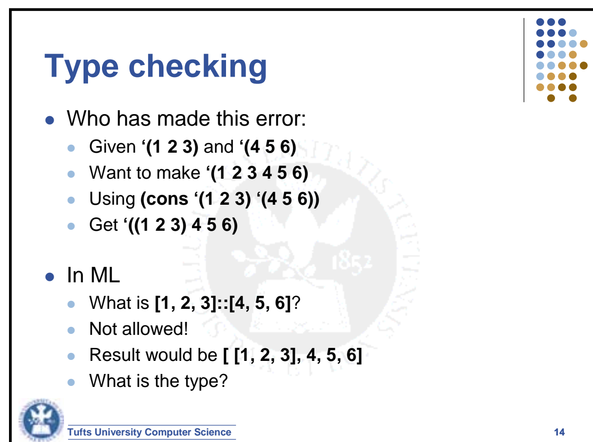


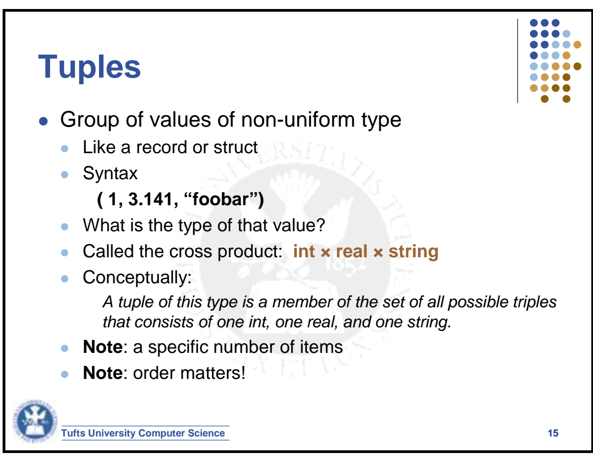
ML functions
Scheme:
(define double (lambda (x) (* 2 x)))
Or shorthand:
(define (double x) (* 2 x))
ML
fun <name> <arg1> … <argN> = <expression>;
Examples:
fun double x = 2 * x;
fun coins q d n p = (q*.25) + (d*.10) + (n*.05) + p;
fun doubleadd x y = double x + double y;
Note precedence: function call higher than addition
Tufts University Computer Science
11

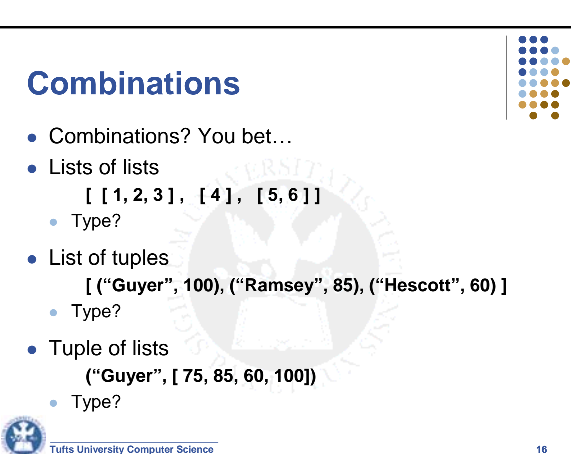


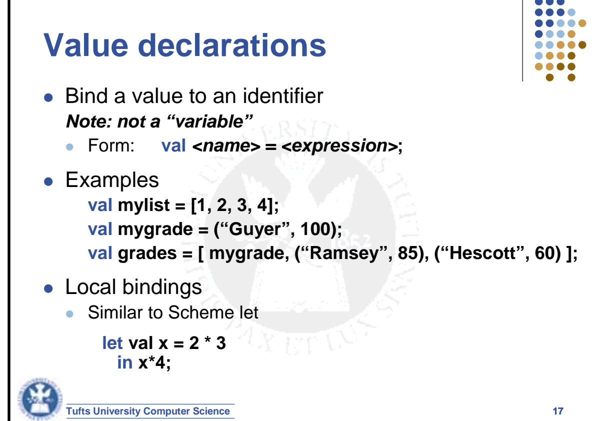
Lists
Similar to Scheme
Different syntax
[1, 2, 3, 4]
What else do you think is different about ML lists?
What is the type of [1, 2, 3, 4]?
[1, 2, 3, 4] : int list
ML lists must contain elements of the same type
What if they didn't?
We could not check at compile-time that computations on the list (e.g., adding up the elements) were legal
Examples…
Tufts University Computer Science
12

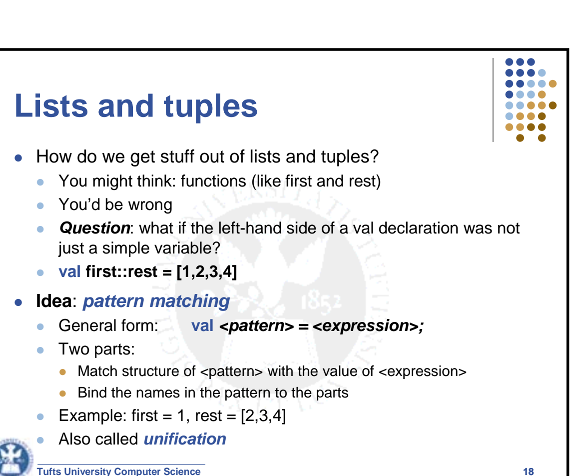

List operations
Scheme
How do we build a list in Scheme?
Cons
ML has cons
Infix operator ::
Scheme: (cons 1 '(2 3 4)) → '(1 2 3 4)
ML: 1::[2, 3, 4] → [1, 2, 3, 4]
Scheme empty list: '()
ML empty list: nil
Scheme: '(1 2 3) = (cons 1 (cons 2 (cons 3 '())))
ML: [1, 2, 3] = 1::2::3::nil
Tufts University Computer Science
13

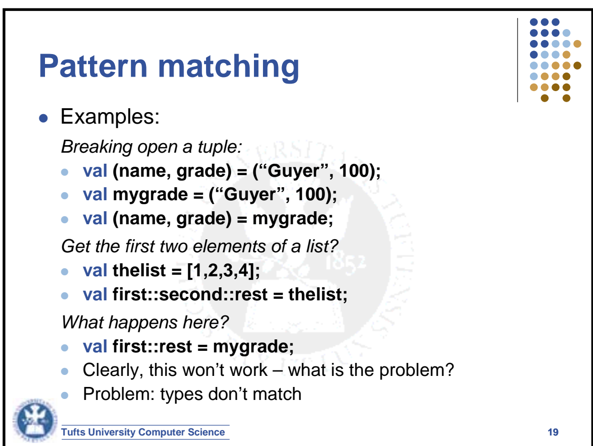

Type checking
Who has made this error:
Given '(1 2 3) and '(4 5 6)
Want to make '(1 2 3 4 5 6)
Using (cons '(1 2 3) '(4 5 6))
Get '((1 2 3) 4 5 6)
In ML
What is [1, 2, 3]::[4, 5, 6]?
Not allowed!
Result would be [[1, 2, 3], 4, 5, 6]
What is the type?
Tufts University Computer Science
14


Tuples
Group of values of non-uniform type
Like a record or struct
Syntax
(1, 3.141, "foobar")
What is the type of that value?
Called the cross product: int × real × string
Conceptually:
A tuple of this type is a member of the set of all possible triples that consists of one int, one real, and one string.
Note: a specific number of items
Note: order matters!
Tufts University Computer Science
15

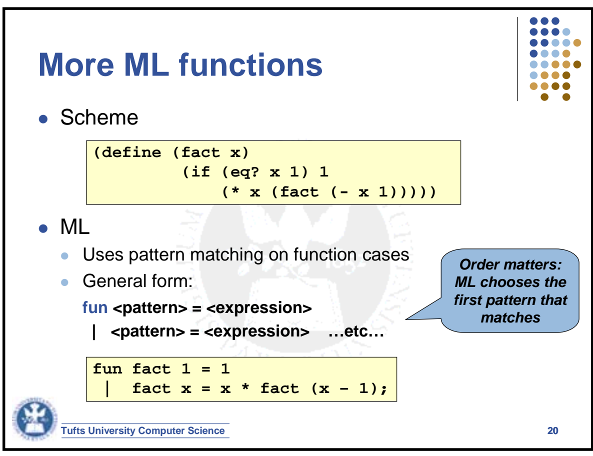

Combinations
Combinations? You bet…
Lists of lists
[[1, 2, 3] , [4] , [5, 6]]
Type?
List of tuples
[("Guyer", 100), ("Ramsey", 85), ("Hescott", 60)]
Type?
Tuple of lists
("Guyer", [75, 85, 60, 100])
Type?
Tufts University Computer Science
16


Value declarations
Bind a value to an identifier
Note: not a "variable"
Form: val <name> = <expression>;
Examples
val mylist = [1, 2, 3, 4];
val mygrade = ("Guyer", 100);
val grades = [mygrade, ("Ramsey", 85), ("Hescott", 60)];
Local bindings
Similar to Scheme let
let val x = 2 * 3
in x*4;
Tufts University Computer Science
17

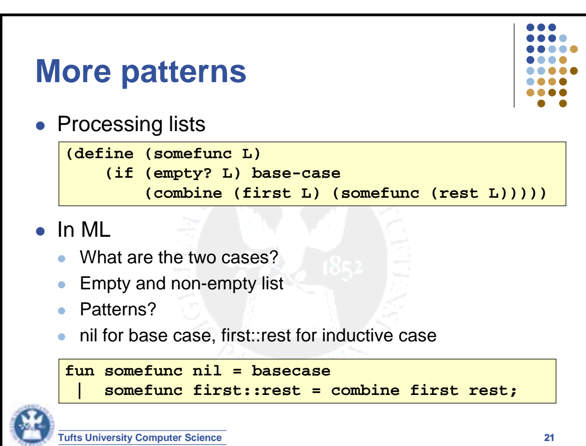

Lists and tuples
How do we get stuff out of lists and tuples?
You might think: functions (like first and rest)
You'd be wrong
Question: what if the left-hand side of a val declaration was not just a simple variable?
val first::rest = [1,2,3,4]
Idea: pattern matching
General form: val <pattern> = <expression>;
Two parts:
Match structure of <pattern> with the value of <expression>
Bind the names in the pattern to the parts
Example: first = 1, rest = [2,3,4]
Also called unification
Tufts University Computer Science
18



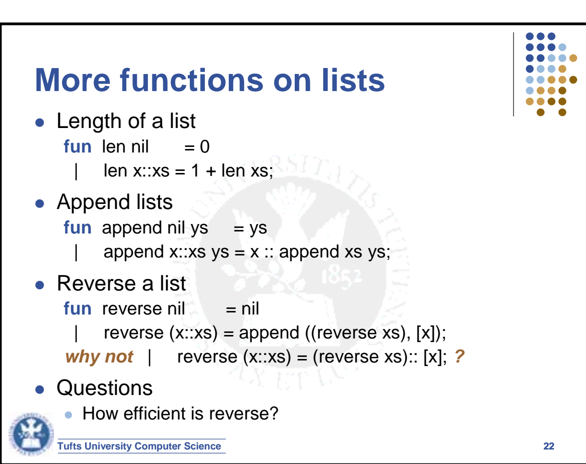
Pattern matching
Examples:
Breaking open a tuple:
val (name, grade) = ("Guyer", 100);
val mygrade = ("Guyer", 100);
val (name, grade) = mygrade;
Get the first two elements of a list?
val thelist = [1,2,3,4];
val first::second::rest = thelist;
What happens here?
val first::rest = mygrade;
Clearly, this won't work – what is the problem?
Problem: types don't match
Tufts University Computer Science
19



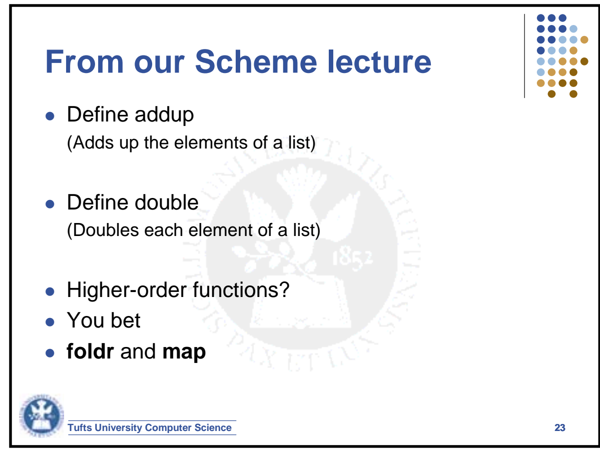
More ML functions
Scheme
(define (fact x)
(if (eq? x 1) 1
(* x (fact (- x 1)))))
ML
Uses pattern matching on function cases
General form:
fun <pattern> = <expression>
| <pattern> = <expression> …etc…
fun fact 1 = 1
| fact x = x * fact (x - 1);
Order matters: ML chooses the first pattern that matches
Tufts University Computer Science
20



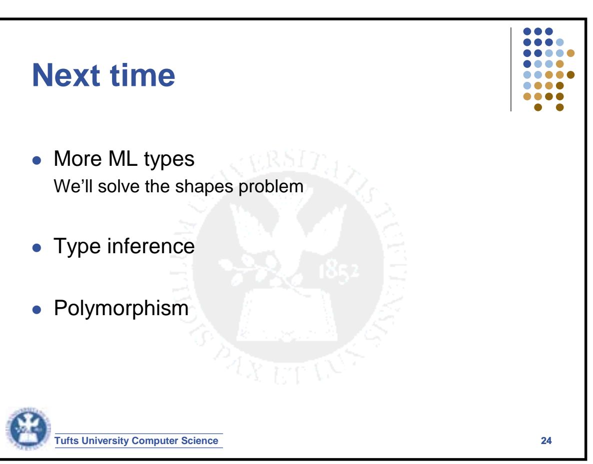
More patterns
Processing lists
(define (somefunc L)
(if (empty? L) base-case
(combine (first L) (somefunc (rest L)))))
In ML
What are the two cases?
Empty and non-empty list
Patterns?
nil for base case, first::rest for inductive case
fun somefunc nil = basecase
| somefunc first::rest = combine first rest;
Tufts University Computer Science
21



More functions on lists
Length of a list
fun len nil = 0
| len x::xs = 1 + len xs;
Append lists
fun append nil ys = ys
| append x::xs ys = x :: append xs ys;
Reverse a list
fun reverse nil = nil
| reverse (x::xs) = append ((reverse xs), [x]);
why not | reverse (x::xs) = (reverse xs):: [x]; ?
Questions
How efficient is reverse?
Tufts University Computer Science
22



From our Scheme lecture
Define addup
(Adds up the elements of a list)
Define double
(Doubles each element of a list)
Higher-order functions?
You bet
foldr and map
Tufts University Computer Science
23



Next time
More ML types
We'll solve the shapes problem
Type inference
Polymorphism
Tufts University Computer Science
24