CSE 341
Lecture 11 a

record types
Ullman 7.1

slides created by Marty Stepp
http://www.cs.washington.edu/341/
Records (7.1.1)

\{fieldName=value, \ldots, fieldName=value\}

- essentially an object; mapping from field names to values

- Example:

  - `val myCar = {make="Toyota", model="Camry", year=1999};`

  `val myCar = {make="Toyota",model="Camry",year=1999} : {make:string, model:string, year:int}`
#fieldName(recordName)

- Example:
  - `#make(myCar);`
    ```
    val it = "Toyota" : string
    ```
  - `#year(myCar);`
    ```
    val it = 1999 : int
    ```

- Does this code set the make to Ford? What does it do?
  - `#make(myCar) = "Ford";`
fun isCool({make, model, year}) =
    make = "Lexus"
    orelse model = "Prius"
    orelse year < 1969 orelse year > 2009;

fun isCool({make="Lexus", model, year}) = true
| isCool({make, model="Prius", year}) = true
| isCool({make, model, year}) =
    year < 1969 orelse year > 2009;

• a pattern can match a complete record
  ▪ field names must match exactly
  ▪ but order of fields declared does not matter
More complex record patterns

(* Returns which of the two cars is more cool. *)

fun cooler(car1 as {make=m1, model=md1, year=y1},
           car2 as {make=m2, model=md2, year=y2}) =
    if year1 < year2 then car2
    else if year2 < year1 then car1
    else if m1 <> m2 andalso m1 = "Kia" then m2
    else m1;

• when matching multiple records, you can give distinct names to its parameters, and/or use the
Partial record patterns

\{\textit{fieldName} [=\textit{value}], \ldots\} \\

fun isCool({\textit{make}="Toyota", \textit{model}, \textit{year}}) =
    \textit{model} = "Prius" orelse \textit{year} > 2000
| isCool({\textit{make}="Kia", \textit{year}, \ldots}) = \textit{year} > 2009
| isCool({\textit{make}="Lexus", \ldots}) = \text{true}
| isCool({\textit{year}, \ldots}) = \textit{year} > 1960;

• a pattern can also be a \textit{partial match} for a record
  ▪ specify the fields you are interested in, followed by \ldots
Mixing datatypes and records

datatype Beverage = Water
| Coffee of {bean:string, caffeine:bool}
| Wine of {label:string, year:int}
| Beer of {brewery:string};

(* Produces a cafe's price for the given drink. *)
fun price(Water) = 1.50
| price(Coffee({bean, caffeine})) = if caffeine then 3.00
| price(Wine({label, year})) = if year < 2009
| price(Beer({brewery})) = 4.00;

- price(Coffee({bean="dark roast", caffeine=true}));
val it = 3.0 : real
Tuples are records (7.1.3)

• A tuple is *syntactic sugar* for a record with number fields

• tuple \((a, b, c)\) is same as record \(\{1=a, 2=b, 3=c\}\)
  - recall: \#1(myTuple), etc.

• in ML they really are interchangeable...
  - `val r = \{1=27, 2=19\};`
  - `val r = (27,19) : int * int`
  - `Int.max r;`
  - `val it = 27 : int`