Pattern Matching

Nate Foster
Spring 2019

Today’s music: *Blank Space* by Taylor Swift
I could show you incredible things // Magic, madness, heaven, sin
So it's gonna be forever // Or it's gonna go down in flames //
You can tell me when it's over // If the high was worth the pain
Textbook vs. lecture
Search

- Website
- Textbook
- Discourse
Section & lecture attendance

about 1% each: incentive not penalty
Review

Previously in 3110:
• Expressions, definitions
• Scope
• Functions

Today:
• Data types for collections: lists, records, tuples
• Pattern matching
## Collections

<table>
<thead>
<tr>
<th></th>
<th>Homogenous elements</th>
<th>Heterogenous elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed number of elements</td>
<td>Lists</td>
<td>Tuples and records</td>
</tr>
<tr>
<td>Unbounded number of elements</td>
<td>Lists</td>
<td></td>
</tr>
</tbody>
</table>
LISTS
List implementation

• **Immutable:** can't change elements
• **Singly-linked:**
  – Good for sequential access of short-to-medium length lists (say, up to 10k)
  – Data structures are tools: none is perfect
• **Terminology:** nil and cons (from Lisp)
• Next lecture: we'll see the implementation
RECORDS
Records vs. Tuples

by name vs.

by position
Records and tuples

• New kind of definition: type definition
• New kinds of types: record types, tuple types
## Collections

<table>
<thead>
<tr>
<th></th>
<th>Homogenous elements</th>
<th>Heterogenous elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed number of elements</td>
<td></td>
<td>Tuples and records</td>
</tr>
<tr>
<td>Unbounded number of elements</td>
<td>Lists</td>
<td></td>
</tr>
</tbody>
</table>

Syntax and semantics: all in the textbook
PATTERN MATCHING
Deconstructing data

Data:

Patterns:
Deconstructing data

Pattern does not match data
Deconstructing data

<table>
<thead>
<tr>
<th>&quot;Andrew&quot;</th>
<th>&quot;CS&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>false</td>
</tr>
</tbody>
</table>

Pattern does match data
Extracts gpa
PATTERN MATCHING ON LISTS
Pattern matching

• Match shape of data
• Extract part(s) of data

Syntax:

```haskell
match e with
| p1 -> e1
| p2 -> e2
| ... 
| pn -> en
```

\(p1..pn:\) pattern expressions
Semantics of pattern matching

• [ ] matches [ ] and nothing else
• h :: t
  – matches 2::[ ], binding h to 2 and t to [ ]
  – matches 1::3::[ ], binding h to 1 and t to 3::[ ]
• _ matches everything
  underscore character, called wildcard
  (it’s like a blank space)

Full details in textbook
Why pattern matching is INCREDIBLE

1. You can’t forget a case
   (inexhaustive pattern-match warning)

2. You can’t duplicate a case
   (unused match case warning)

3. You can’t get an exception
   (e.g., \texttt{hd [ ]})

4. Pattern matching leads to elegant, concise, beautiful code
Upcoming events

- [Thu] Foster Office Hours
- [Fri] First recitations due on CMS

This is incredible.

THIS IS 3110