LIVING COMPUTATION

WEEK 2/17

CS491.2/591.2 Fall 2017
UNM Computer Science

M,W 11:00-12:15 CEC-B146A

Dave Ackley
ackley@cs.unm.edu
→ Notes to wiki at https://robust.cs.unm.edu/

Questions and comments

Status of robust.cs.unm.edu

No class next Monday!

Look for ulam 3.0.6 release before next class!

- Directories-with-hideous-spaces detection
- Improved documentation generation
- Miscellaneous bug fixes
1. Report in (All, 10 min): Q&A&A

2. Show and tell (All, 20 min): Some Cool Elements

3. Break (All, 5 min): (Extra demos ad lib)

4. Tell (Dave, 35 min): Intro to ulam Part 12: Quarks and inheritance, data members and the bit budget

5. Report out (Dave, 5 min): Action items.
The two attractors of computing

<table>
<thead>
<tr>
<th>Canonical mechanism</th>
<th>Algorithm</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admissibility requirement</td>
<td>Correct</td>
<td>Robust</td>
</tr>
<tr>
<td>Primary goal</td>
<td>Efficient</td>
<td>Correct</td>
</tr>
<tr>
<td>Secondary goal</td>
<td>Robust</td>
<td>Efficient</td>
</tr>
</tbody>
</table>

Attractor nickname

- "Master of the Universe"
- "Member of the Team"
The two attractors of computing

<table>
<thead>
<tr>
<th>Canonical mechanism</th>
<th>Algorithm</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admissibility</td>
<td>Correct</td>
<td>Robust</td>
</tr>
<tr>
<td>requirement</td>
<td>Efficient</td>
<td>Correct</td>
</tr>
<tr>
<td>Primary goal</td>
<td>Robust</td>
<td>Efficient</td>
</tr>
<tr>
<td>Secondary goal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computational structure</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attractor nickname</td>
<td>“Master of the Universe”</td>
<td>“Member of the Team”</td>
</tr>
</tbody>
</table>
Let's look at three today
Volunteers or volunteered?
(Extra demos ad lib)
ulam: The Event Window

Movable Feast Machine
Event Window
Coordinate Systems

Manhattan distance
0
1
2
3
4
≥5

© 2016
Dave Ackley
http://creativecommons.org/licenses/by/4.0/
Data members

- Elements can have data members
- Data members are packed in lexical order, with no padding between them
- Total space available for element data members: 71 BITS. (Why such a weird number?)
- What's the difference between:

  ```
  element Foo {
    EventWindow ew;
    Void behave() {
      ew[1] = ew[0];
    }
  }
  
  element Foo {
    Void behave() {
      EventWindow ew;
      ew[1] = ew[0];
    }
  }
  ```

- (Demos)
• Classes smaller than elements that can be used as parts of elements
  – Why do they exist? In most OO languages it's just 'objects all the way down'.
  – Not only smaller than elements but max 32 bits

• Examples from 'stdlib', the ulam standard library
  – EventWindow, Random, Fail
  – Once
  – (In /usr/lib/ulam/ULAM/share/ulam/stdlib )

• Demo: ID
Inheritance

- ulam supports single inheritance
  - Foo inherits from Bar: `element Foo : Bar { .. }`
  - Bar inherits from Gah: `quark Bar : Gah { .. }`
  - What does Gah inherit from?
    - All inheritance is 'public'
- Examples from the ulam 'stdlib'
  - UrSelf
- Demo: QID
Self and Super

- Predefined class typedefs:
  - Self
  - Super
- E.g., instead of writing:
  ```quark Foo {.. if (ew[1] is Foo) ..}
prefer:
    quark Foo {.. if (ew[1] is Self) ..}
```
- Predefined class references
  - self - a Self reference to the current object (like 'this' in Java/C++)
  - super - a Super reference to the current object
  - Can call overridden methods via super!
0. Create a page for yourself at robust.cs.unm.edu. (You log in to the wiki with your CS id/pwd.)

1. Ensure the public key for your default handle is on that page!

2. Work on ulam exercises 11. Send me an .mfz file (not a tar file, not loose ulam files, etc) by 10AM Wednesday Sep 6 2017!
Work on the first two anyway. The third one, if attempted, may be easier using the 'C2D' class from stdlib.

1. **Majority.ulam:** Complete the supplied Majority.ulam class as described in its class comment. (Find it at http://www.cs.unm.edu/~ackley/591/code/20170830-code/)

2. **QCool.ulam:** Make a quark QCool that does something that you decide! Make at least two elements that inherit from QCool for some sensible reason! Be able to tell us about it!

3. **Check3.ulam:** (Optional) Starting from a single Check3 atom anywhere in an otherwise empty grid, create a space-filling checkerboard consisting of 3x3 blocks of Check3s alternating with 3x3 empty sites! Emphasize clean code!