Homeostatic architectures for robust spatial computing

David H. Ackley
Lance Williams
University of New Mexico
Computer Science

The Robust Physical Computation Group

Spatial Computing Workshop 2011
Ann Arbor, MI
October 3, 2011
Plan

- Advocacy /
  - We have a problem
  - How we got into this mess
  - Robust spatial computing
- Research /
  - The movable feast machine
  - Homeostatic computation
  - Demos
- Call to action / Test to failure
Our security train wreck

• First bug costs the machine

• Non-solutions:
  • Blaming the user / user education
  • Blaming the developer / fixing the last bug

• Solutions:
  • Blame von Neumann
Computation must be born again

Instead of being born again, why not just GROW UP?

► OK, our answer might be wrong
How we got into this mess

Digital hardware is *massively* redundant

One person's analog signal..

..is another person's digital noise.

The Original Deal:

**Hardware shall be reliable**

**Software shall be efficient**

(* Computation shall be serial)
Architecture matters: Space

- CPU + RAM
- Von Neumann’s lament
- Strategy: Let space be space
  - Consequences: Fungibility, scalability
Architecture matters: Time

• Scalability issue: The light cone
• Robustness issue: **Single** source clocking
• Design issue: Who waits for whom?
• **Synchronous design begs the question**
Architecture matters: Correctness

- Who’s kidding whom?
- If not correctness?
- Best effort is better than correct
Indefinite scalability

- A single, clean, architectural criterion implying:
  - Spatial computing
  - Robust computation
- Perhaps a tad ambitious
Indefinite scalability
Let space be space, let time be time

• Sacrificing:
  ✗ Fixed-width addresses, unique node names.
  ✗ Logarithmic global communication cost
  ✗ Single source clocking, phase synchronization
  ✗ 'Times' – run time, load time, power on time..

• Embracing:
  ✔ Opportunistic reproduction for ||ism & robustness
  ✔ Movability for configuration, manifest destiny, ...
  ✔ Multilevel robustness: Up to the end-user
Living Computation

• Impossible working conditions:
  • Program inputs might be late, missing, wrong
  • Program execution might be faulty
• Become livable if
  • Program outputs can be wrong, late, missing
• Because:
  • Others are duplicating/checking your work
• Efficiency and robustness are mortal enemies
An example: Software engineering as artificial chemistry

The Movable Feast Machine

spatial computing
relative addressing
local connectivity
indefinitely scalable
fixed size atoms/sites for mobility
parallel asynchronous update
element-oriented programming

/* DReg: Dynamic Regulator. */

element DReg() = 0xdba {
    if n:anyAt(1), n is Empty, odds(1,1000) then n = DReg;
    if n is Empty, odds(1,200) then n = Res;
    if n is DReg, odds(1,10) then n = Empty; // limit DRegs
    if odds(1,100) then n = Empty;
}
Processing

- Hardware packs as many disjoint event windows into space-time as possible.

- Typical parameters:
  - 64 bits per site; 16 bit header + 48 bonds and/or state
  - Event window radius 4 "L0" distance
  - Bonds are symmetric and relative

- Software defines a set of types with atomic formats and behavioral rules; initial conditions
Movable feast machine

Sites & atoms
Movable feast machine

Event window
Movable feast machine

Visualization
Demon Horde Sorting: Robust Computation Example

- Task: Flow sort endless data stream
  - It’s impossible
- 'Maxwell's Demon' sorting elements maintained in homeostasis by DReg
- Surprise: Quality vs data rate.
Call to action

Computation shall be

As robust as possible
As efficient as necessary
As correct as a Google search