In Pursuit of a New Kind of Computer

Programable Smart Machine Lab (PSML)
Jonathan Appavoo, Boston University

“And now for something completely different...”
Properties

- Simply and yet Richly Programmed
Properties

• Simply and yet Richly Programmed
• Automatically improves with its size
Properties

- Simply and yet Richly Programmed
- Automatically improves with its size
- Automatically improves/adapts with experience
Properties

• Simply and yet Richly Programmed
• Automatically improves with its size
• Automatically improves/adapts with experience
• Amenable to implementation with low power devices
Properties

• Simply and yet Richly Programmed
• Automatically improves with its size
• Automatically improves/adapts with experience
• Amenable to implementation with low power devices

A programmable child that is pedantically obedient without the attitude :-}
Simply and yet Richly Programmed

Let's not throw the baby out with the bath water

Simple synchronously clocked uni-processor system

- Easy to grok deterministic model
- General Purpose 'elastic'
- Easy to implement higher level SW machines

First Draft of a Report on the EDVAC
by
John von Neumann

Contract No. W-670-ORD-4926
Between the
United States Army Ordnance Department
and the
University of Pennsylvania
Hmmm Now What?

- Simply and yet Richly Programmed
- Automatically improves with its size
- Automatically improves/adapts with experience
- Amenable to implementation with low power devices
Hmmm Now What?

- Simply and yet Richly Programmed
- Automatically improves with its size
- Automatically improves/adapts with experience
- Amenable to implementation with low power devices

Using knowledge of runtime behavior introduce a cache based optimization:

```c
hash table ht

get(key,value)  put(key,value)

key = hf(s);
if (get(key,&value)==hit) {
    fast(value);
} else {
    slow(&value);
    put(key,value);
}
```
Hmmm Now What?

- Simply and yet Richly Programmed
- Automatically improves with its size
- Automatically improves/adapts with experience
- Amenable to implementation with low power devices

Yikes...
I did warn you ;-)
3 \times 7
Need an Interface

Logic Based Computer

ALU

CLOCK

IO

MEMORY

Stimulus Signal

Extract

Response Signal

Apply

Learning System (eg. Associative Memory, Neural Network, Deep Network)

Extract a Signal
Execution Signal

m-bit binary state vector $s$
(registers + ram + I/O channels)

```c
loop(void) {
    while(1){
        fetch(s, &op);
        execute(op, &s);
    }
}
```

trace of execution $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_{i-1} \rightarrow s_i \rightarrow s_{i+1} \rightarrow \ldots \rightarrow s_n$

See our *HotPAR'12* Paper, "Parallelization by Simulated Tunneling", Waterland et al. For a more technical view of execution as state space traversal (Dynamical Systems Interpretation)
A 524352-bit binary vector $s$ forms the entire 6502 system state. Each step $i$ of execution produces a vector $s_i$. 
i

i+1

i \text{ xor } i+1
Intriguing Possibilities

- “Derivatives” expose fascinating structure
- Simple to construct and study filters
- Unified statistical representation exposes unexpected patterns across all of system execution
- Opportunities for studies are proving to be amazingly fun and challenging our intuitions
Need a Method

How can we Apply the Response

Logic Based Computer

ALU

CLOCK

MEMORY

Extract

Stimulus Signal

Apply

Response Signal

Learning System (eg. Associative Memory, Neural Network, Deep Network)

2

How can we Apply the Response
m-bit binary state vector \( s \)
(registers + ram + I/O channels)

```
loop(void) {
  while(1){
    fetch(s,&op);
    execute(op,&s);
  }
}
```

State Pairs are computation (Again See our *HotPAR'12 Paper*)
m-bit binary state vector $s$ (registers + ram + I/O channels)

```
loop(void) {
    while (1) {
        fetch(s, &op);
        execute(op, &s);
    }
}
```

State Pairs are computation (Again See our *HotPAR'12* Paper)
Putting it together

m-bit binary state vector \( s \)
(registers + ram + I/O channels)

fetch
update
execute
look-up
loop

\[ s_0 \ s_1 \ s_2 \ldots \ s_{i-1} \ s_i \ s_{i+1} \ldots \ s_n \]
trace of execution

Cache of state pairs
Predictive Execution
State Pairs
Predicted States

Learning System
(eg. Associative Memory, Neural Network, Deep Network)
Sorry Animated gif broken in PDF VERSION
Hints of Smoke

- Deterministic computation
- All I/O up front
- Restricted x86 simulator
- MPI on Blue Gene
- Simple learning hypothesis
- Simple Bayesian predictor

Initial results are documented again in our HotPAR'12 Paper.
Computational Battery

- Split things up into two distinct phases a learning problem (charging) and later use mode
- train ML model on large scale systems and store model along with initial DB of state pairs
- VMs uses this along with local cores to accelerate computation
All Done the Talk ... Work is Just Starting

- These are all just first steps and all very rough
- But wow a ton of fun!
- Able to apply and explore fascinating relationships between classical logic and statical mechanisms
- Amos and I have been growing the set of crazies (from complexity, information theory, physics, mathematics, and HW)
- Thanks to them all: Margo Seltzer, Steve Homer, and all the brave and excellent students that have joined Amos: Katherine Zhao, Elaine Angelino, and others.