Wavenets

John Chapin
MIT
\[ A \rightarrow B \]

**Diagram: A**

1. Node A
2. Edge Z
3. \[ 1:z + 2:s \rightarrow 3:t \]

**Diagram: B**

1. Node A
2. Node B
3. \[ 1:z + 2:s \rightarrow 3:t \]

**Diagram: C**

1. Node A
2. Edge Z
3. \[ 1:z + 2:s \rightarrow 3:t \]

**Diagram: D**

1. Node A
2. Node B
3. \[ 1:z + 2:s \rightarrow 3:t \]
Wavenet = Synchronous circuit
+ Superposition
+ Wave function collapse
+ Conservation of energy
Wavenet =
A simple primitive that provides:
Global concurrency control
Automatic resource management

Dining philosophers solution
p:think => left:fork + right:fork
p:hungry + left:fork + right:fork => p:eat
p:hungry => p:tryagain

(right:fork => table:fork)
(left:fork => table:fork)
(table:fork => right:fork)
(table:fork => left:fork)
(table:fork --> table:fork)

As many philosophers and forks as desired

The challenge
Computing the latching is NP-complete
Conservation of energy =
A powerful primitive that solves many different problems
Efficient implementation?
Offline analysis
Hardware support
Why?

Batch programs  Parallel servers
Input  Run forever
Processing  Maintain invariants
Output  Respond to multiple
Halt  inputs in parallel
Turing machines  Wavenets
Lambda calculus  Calculus?
Procedures  Prog. abstraction?

Take off the Turing blindfold!

http://sdg.lcs.mit.edu/wavenets