Clocks

Adam T. Sampson
School of Computing, University of Kent

Neil C. C. Brown
School of Computing, University of Kent

Motivation

My day job

- Big concurrent distributed simulations
- Lots of agents
- Discrete time steps
  - Implemented using a barrier
  - Each step divided into several phases

That's an awfully nice wheel

- Event-based simulation is very similar
- Actions occur at particular times
- Implement as priority queue of actions
  - Sorted by the time at which the action should occur

Barrier

- Has:
  - Enrollment count
  - Set of blocked processes
- When count == size(set), schedule all in set

Clock

- Has:
  - Enrollment count
  - Priority queue of blocked processes
- When count == size(queue), schedule all the processes that have the lowest time

Agent Smith

```
WHILE alive
   ...  query environment
   ...  think
   ...  write changes back to environment

   WHILE now < next.time
     SYNC bar
```

Agent Orange

```
WHILE alive
   ...  query environment
   ...  think
   ...  write changes back to environment

   SYNC clock, next.time
```

- This runs faster – because it doesn't have to wake up on timesteps where it's not doing anything (which is most of them)

Some questions

- What should the semantics for ALT be?
- Is it worth having language bindings?
  - This can just be implemented as a process...

Any questions?

- Clocks are implemented in CHP – go and play!