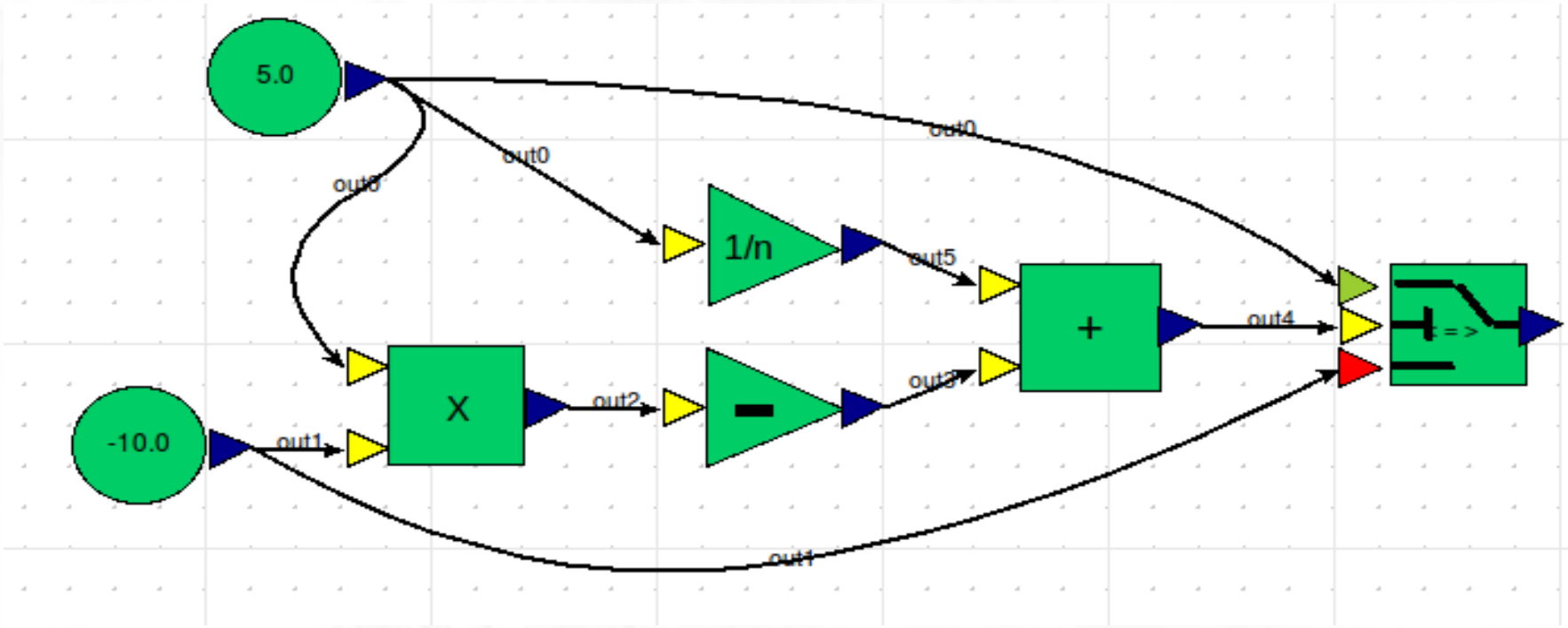
Embedding Causal Block Diagrams Within Behaviour Trees

Bentley James Oakes

Outline

- Problem to Address
- Proposal
  - Formalisms Used
- Experiments
- Results
- Future Work
- Conclusion

Causal Block Diagrams



Blocks compute values at each time step
Outputs connected to inputs

Problem

========== Signals Dumping Begin ==========

multiply0 Product [0.1, -0.0, -0.011, -0.022, -0.032, -0.043]
secondmultiply Product [0.0, 0.11, 0.110, 0.108, 0.106, 0.103]
delay0 Delay [1.0, -0.0, -0.11, -0.22, -0.328, -0.435, -0.539]

========== Signals Dumping End ==========

How do we visualize/connect to simulation?

Proposal

Find formalism to embed causal blocks into

Criteria:

Equivalent

- One-to-one representation?
- Increase in power?

Easily understood/changed

Easy to visualize (model/simulation)

Behaviour Trees

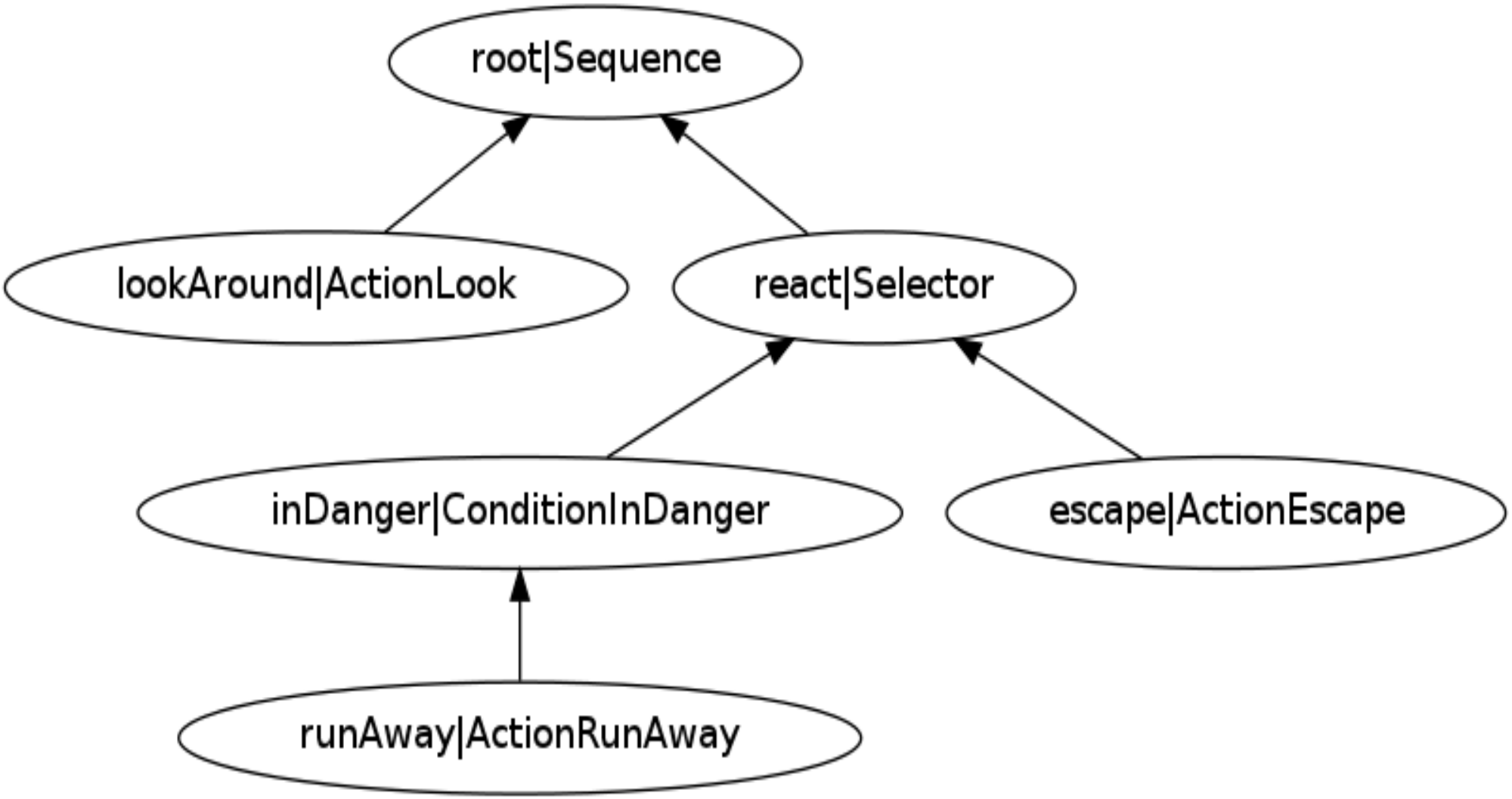
Each agent in a simulation has a tree

Query tree to determine action of agent

Starting at root node:

- Each node will:
  - Query children
  - Perform action
  - Return true/false

Behaviour Tree Example



BT Advantages

Easy to construct/rearrange trees
    Used in video games: Halo 2/Spore

Trees can control animation/sounds/AI
    Merge calculations with higher-level control

Lazy evaluation
    Don't solve everything at once

Similarities to CBD

Both have nice tree structure
- Modular design
- Combine nodes for any function

Values are passed up tree

Exploration of a formal equivalance
- CBD → Behaviour tree
- Behaviour tree → CBD

Differences

Algebraic loops in behaviour tree?
- Special handling required
- Not implemented in project

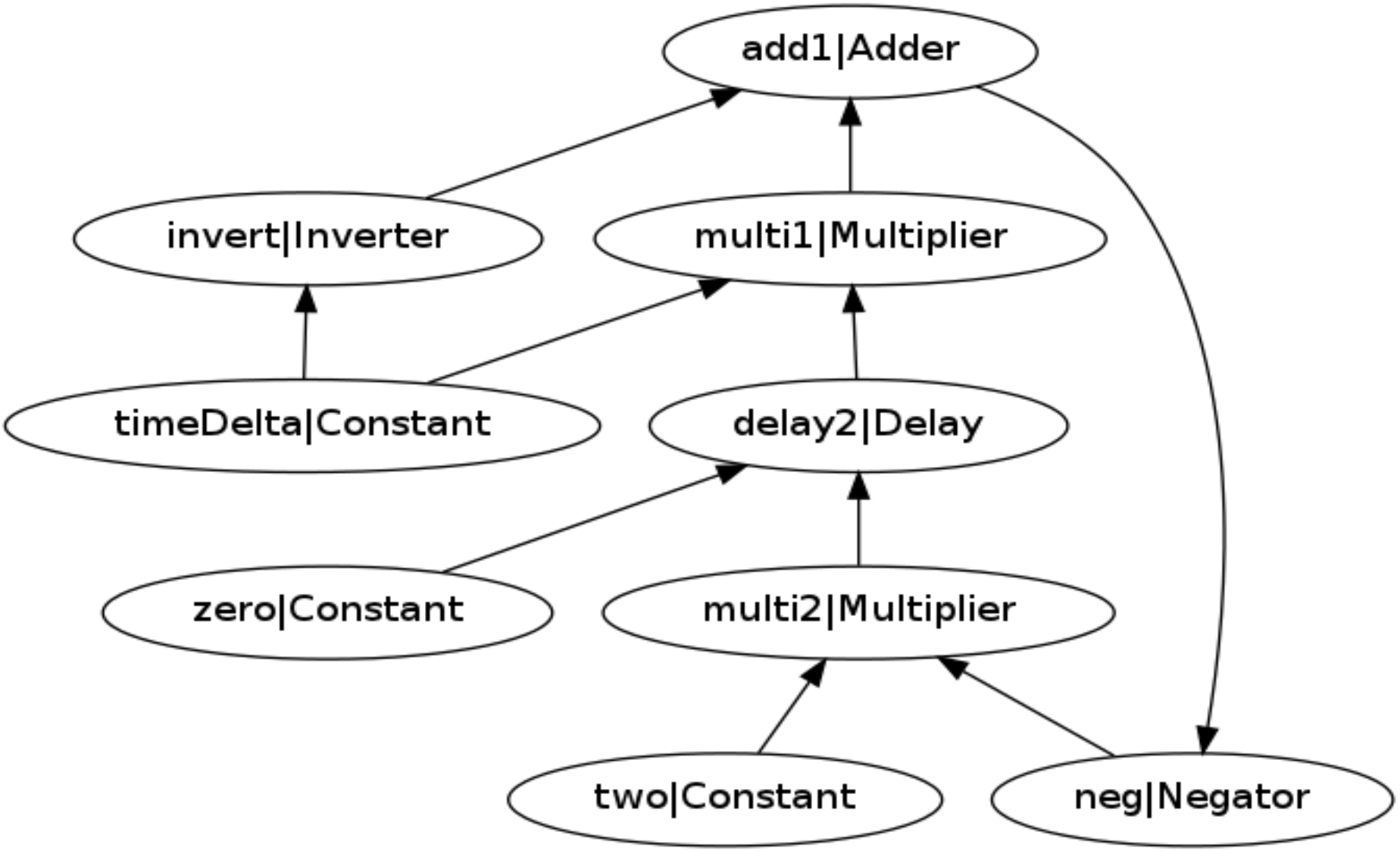
Only true/false value passed up
- Easily fixed → change to float
- 0 is false, anything else is true

Embedding

CBD blocks map onto BH nodes directly

- Adder
- Multiplier
- Inverter
- Negator
- Constant
- Test
- Delay
- Integrator/Derivative*

Sample Embedding



Embedding

Multiplier node update():

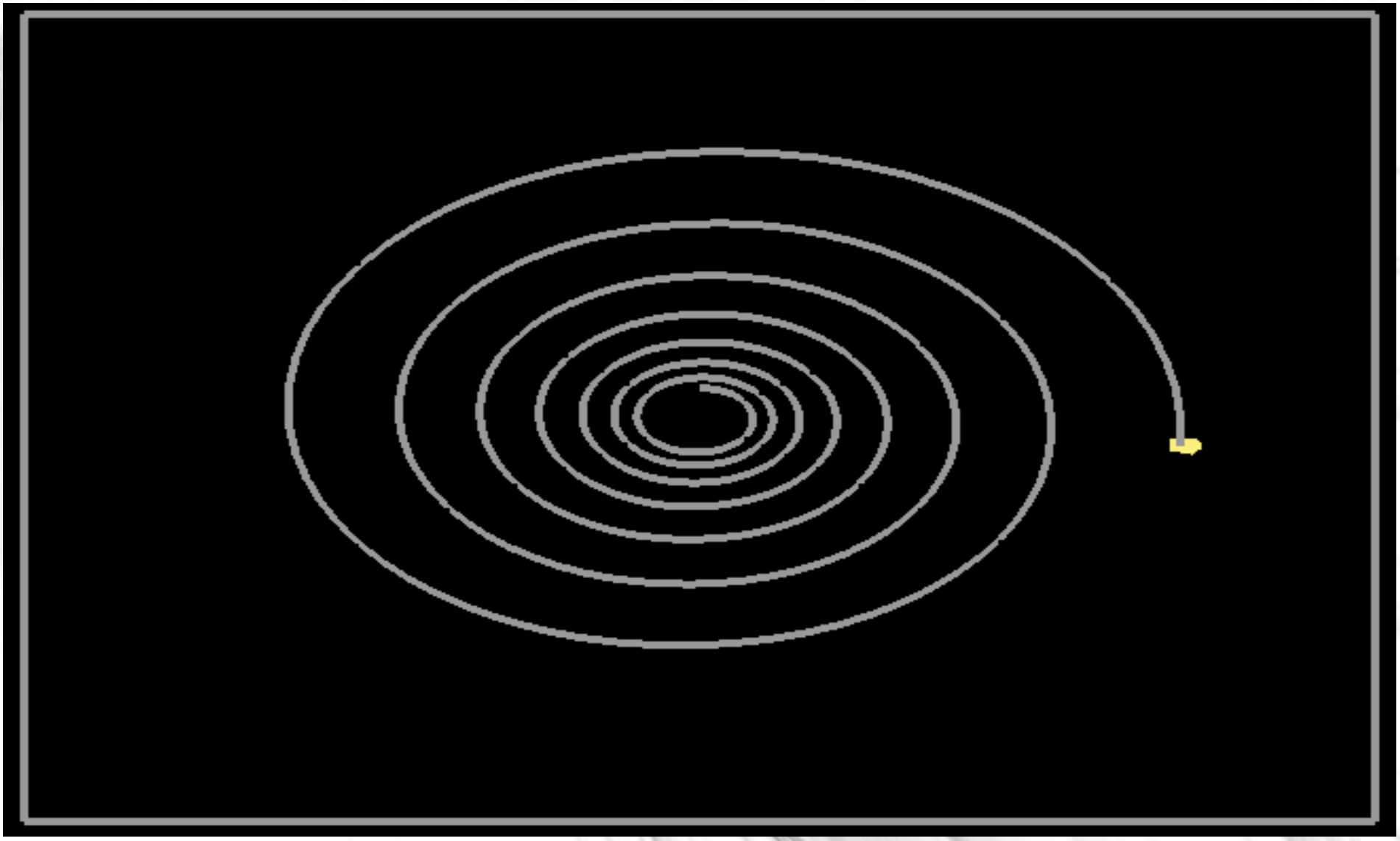
```
If children.size == 0
    output = 0

Result = 1
For Child c in children
    Result *= c.update()

Output = result
```

Experiments

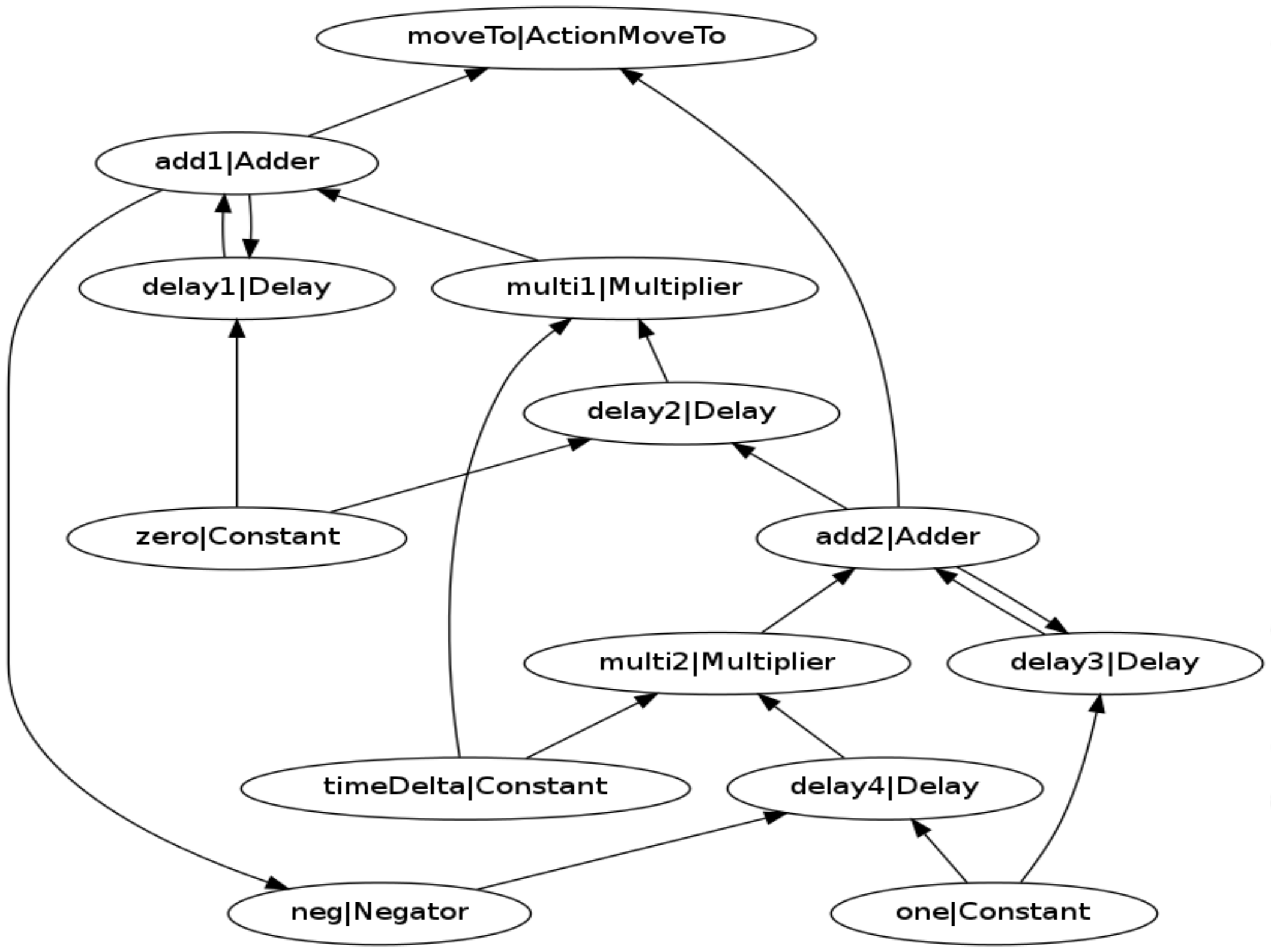
Circle Test:
Use integrators to draw x(t) vs dx/dt(t)

Personal Space:
Many agents in world
One agent walks from start to goal
All agents move away from each other

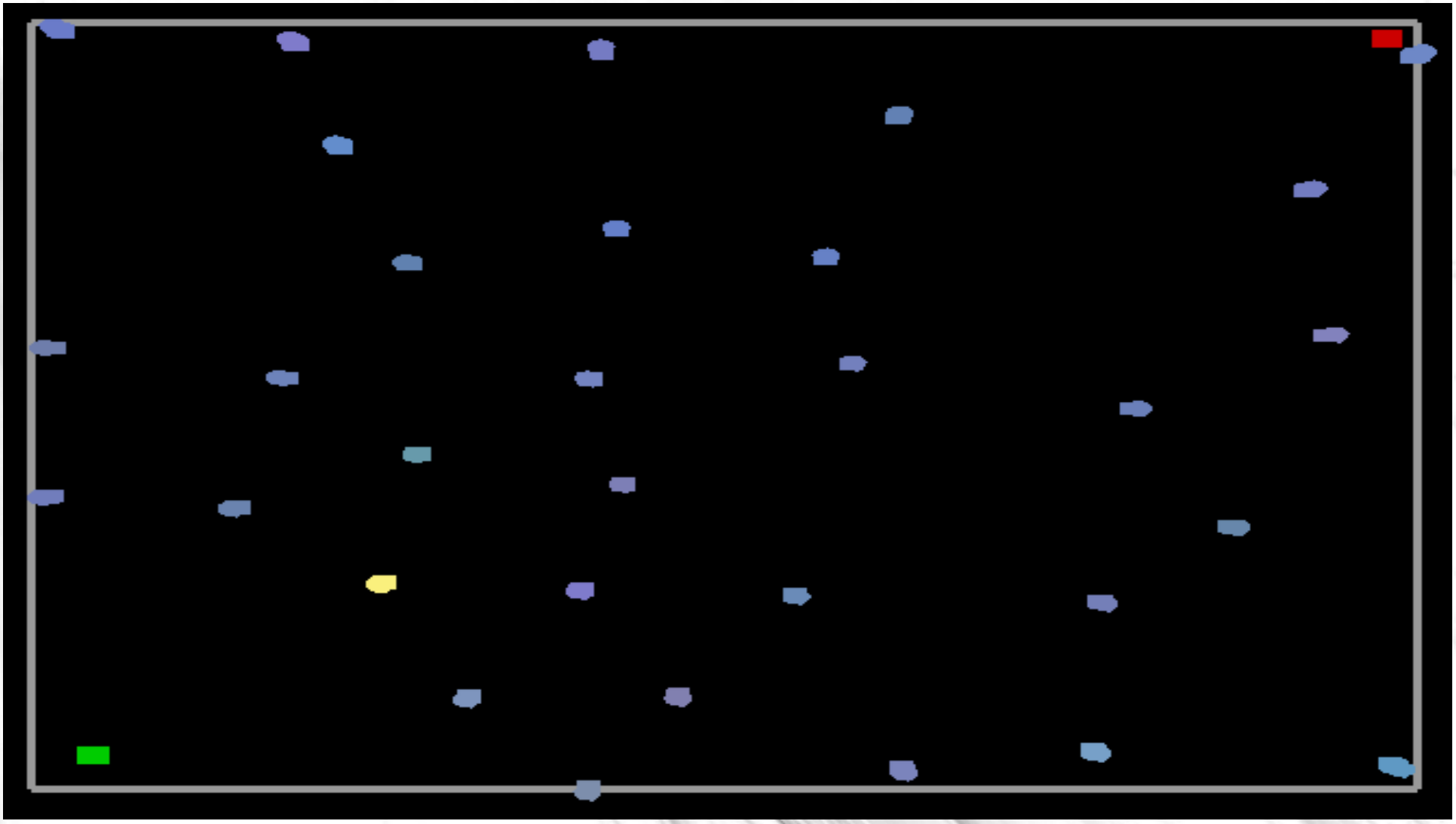
Results – Circle Test



Results – Circle Test


moveTo|ActionMoveTo
add1|Adder
delay1|Delay
multi1|Multiplier
delay2|Delay
zero|Constant
add2|Adder
multi2|Multiplier
delay3|Delay
timeDelta|Constant
delay4|Delay
neg|Negator
one|Constant

Results – Personal Space



Results – Personal Space

Future Work

Detect algebraic loops

Implement behaviour trees in Atom3

Add detail to model visualization

Explore formal equivalence

Conclusion

Behaviour trees add lots of power
- Control flow
- Handle animation/events
- Superset of CBD?

Visualization needed
- Evolution of values
- Hook into simulation

References

Clark Verbrugge. COMP 521 course notes. 2012.

Hans Vangheluwe. COMP 522 course notes. 2012.

Damian Isla. Managing complexity in the Halo 2 AI system. In Proceedings of the Game Developers Conference, 2005.

Chong-U Lim. An A.I. Player for DEFCON: An Evolutionary Approach Using Behavior Trees.
Imperial College, London.

Questions?