Physical Swarm behaviors using evolved Behavior Trees

Liam Koehler
Division of Science and Mathematics

Agents

Many automated processes contain the concept of an agent. Agents are autonomous, self contained units that react to the world in a convincing and predictable way.

Agent Example

An NPC (non-player character) should **fight**, unless it's *health falls too low*, at which point it should **run away** and find a safe space to **heal**.

Agent Example

An NPC (non-player character) should **fight**, unless it's *health falls too low*, at which point it should **run away** and find a safe space to **heal**.

- The **bold** statements represent behaviors
- The *italic* statement represents the control structure used to pick behaviors
- Creating and combining these components is called **agent modeling**, which is a task we can use behavior trees for

Table of contents:

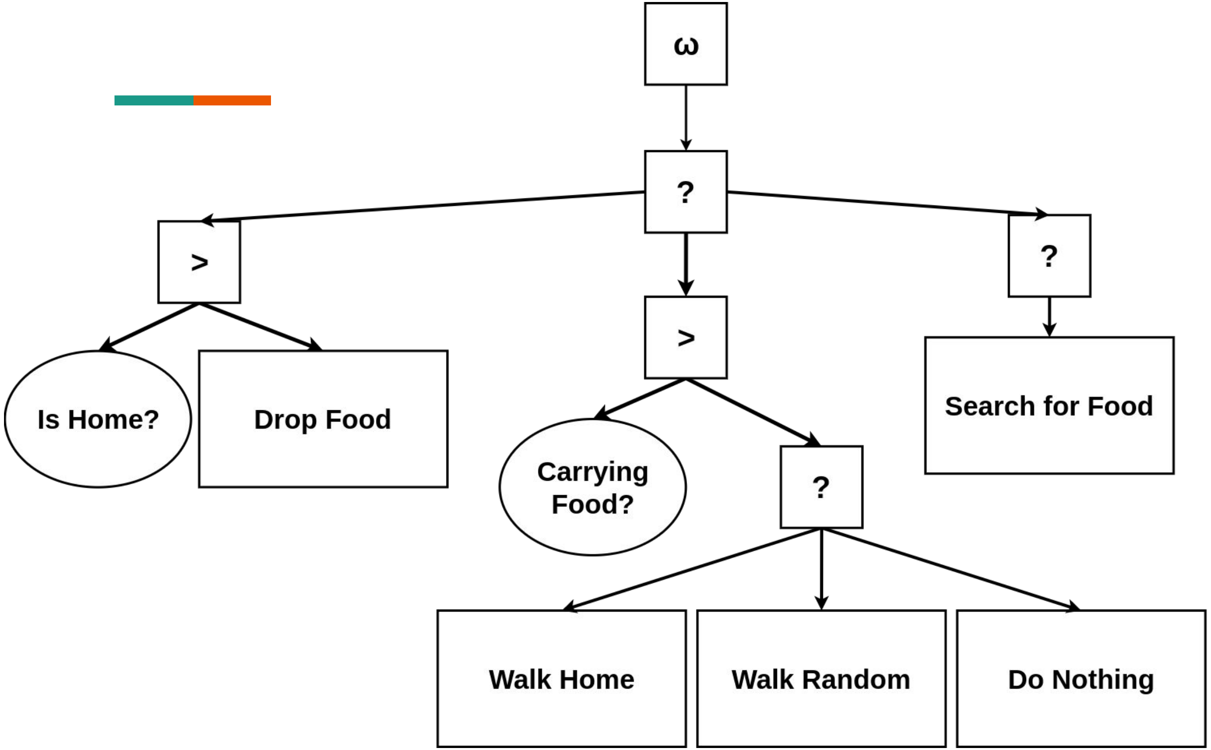
Behavior Trees

Behavior Trees

Behavior Trees are a simple, tree-based structure for modeling agent behavior.

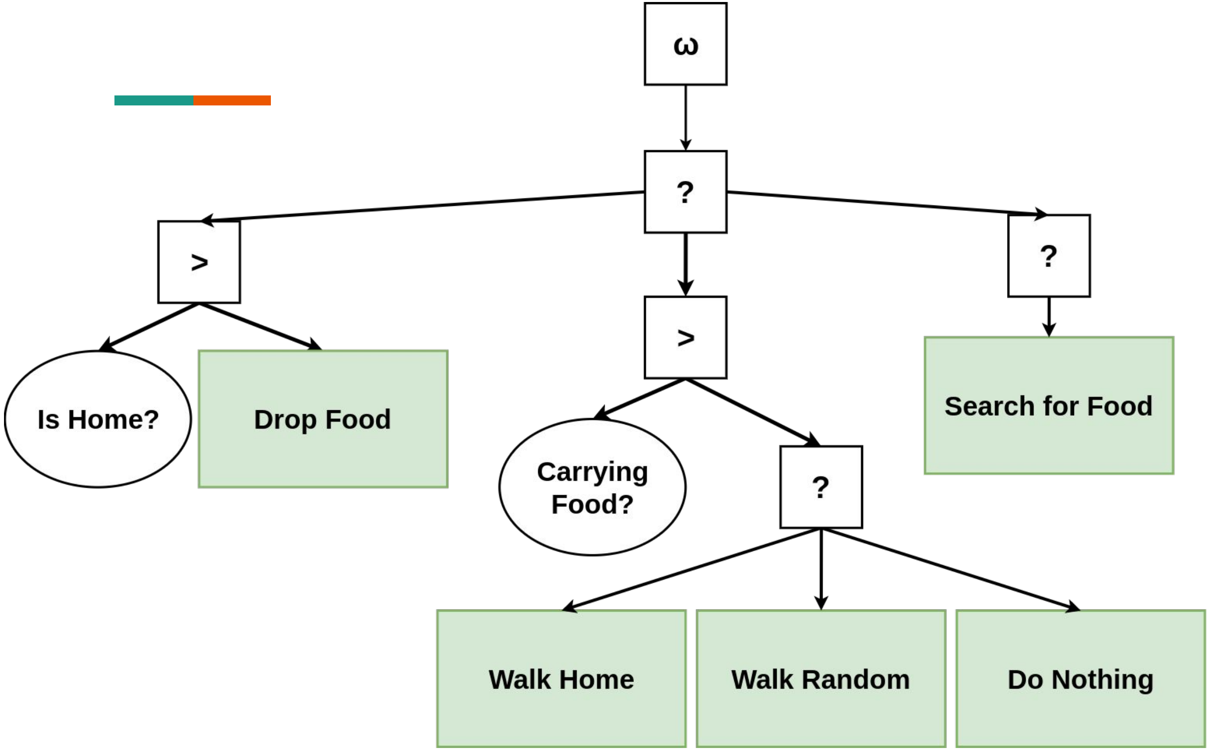
Uses include:

- Video game NPCs
- Robotics
- Swarm Modeling



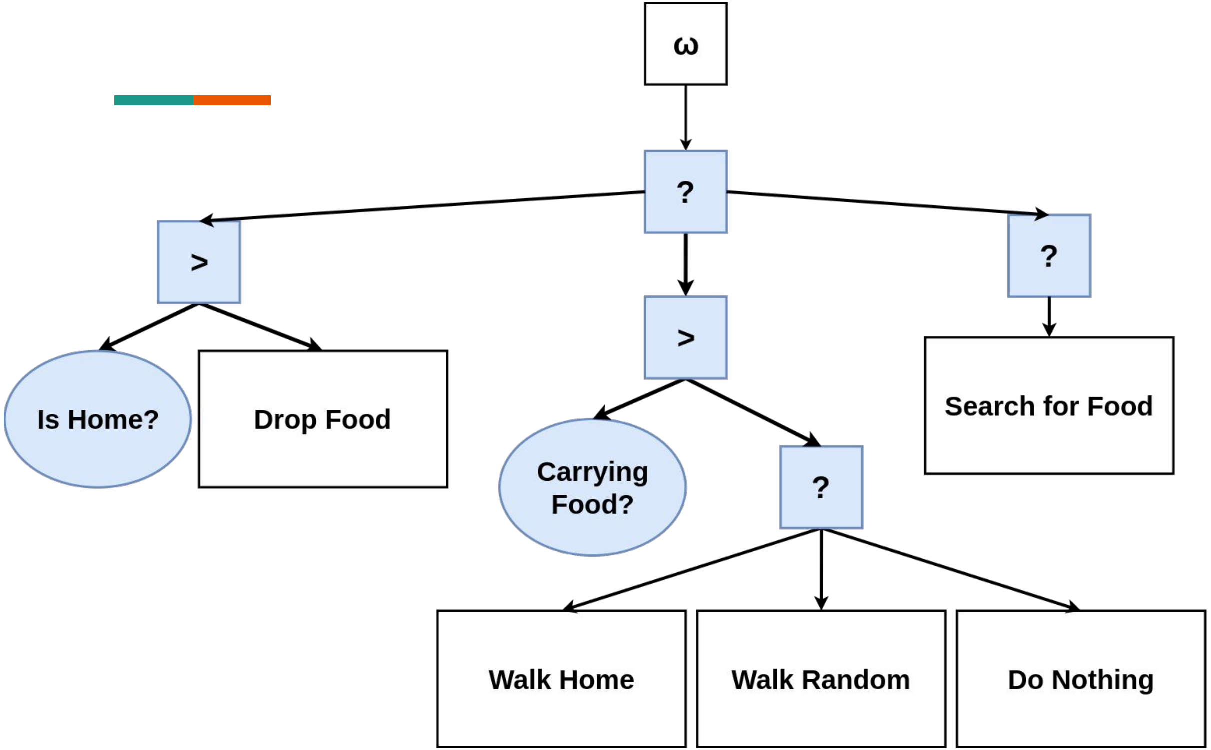
Structure:

- Tree based
- Evaluation starts at the top
- Made up of nodes of various types
- Picks behaviors



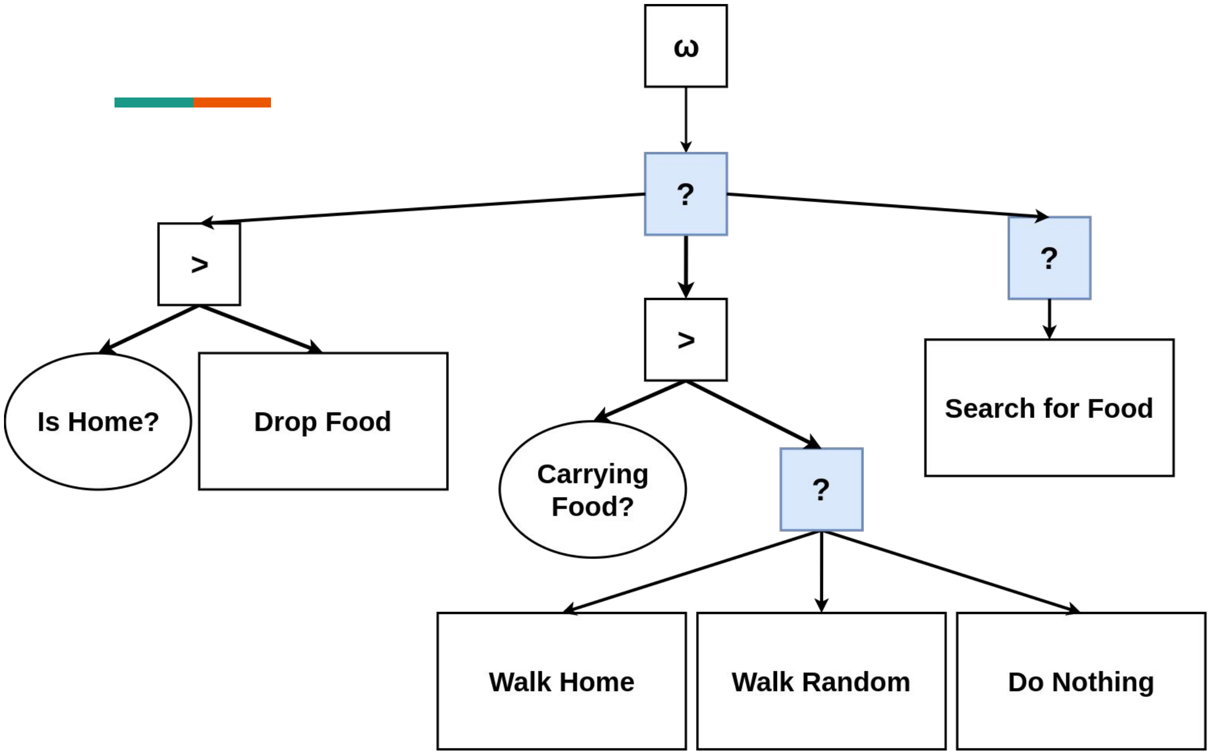
Behaviors:

- Found at the leaves
- Represent tangible actions
- When reached, the behavior will repeat until a new behavior is selected



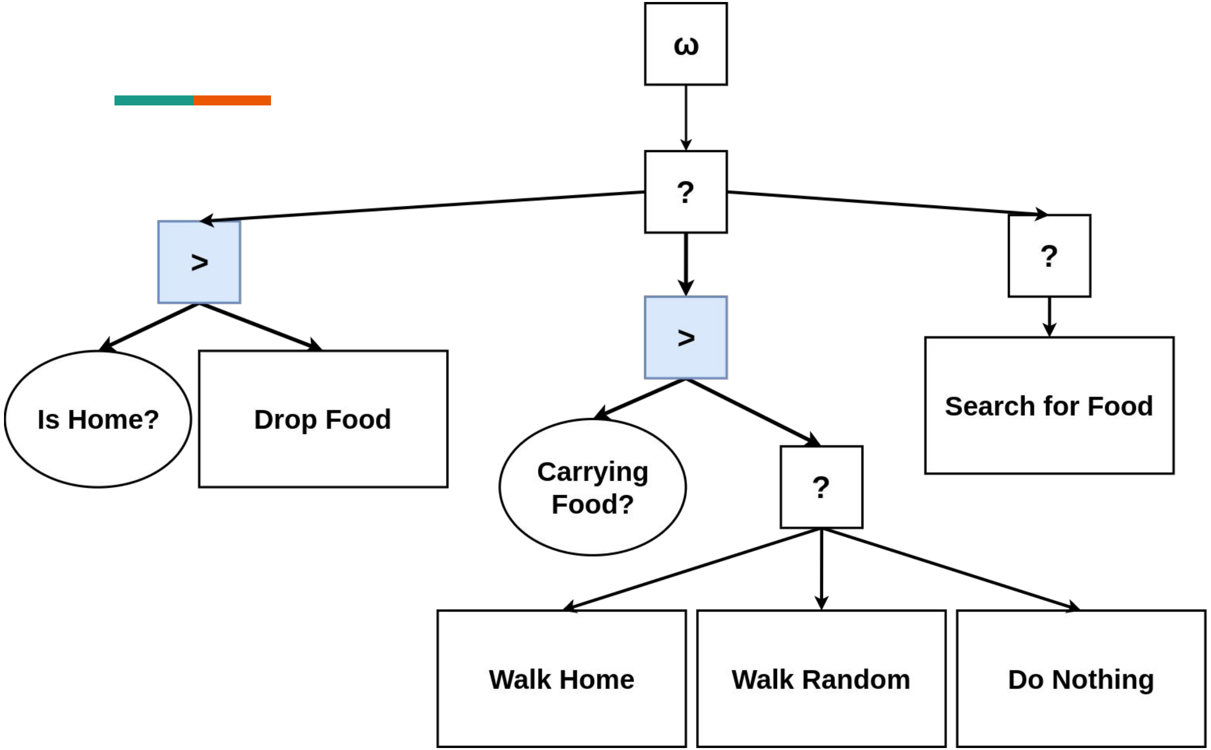
Decision Nodes:

- Represent the control structure
- Must have one parent
- Typically must have at least one child



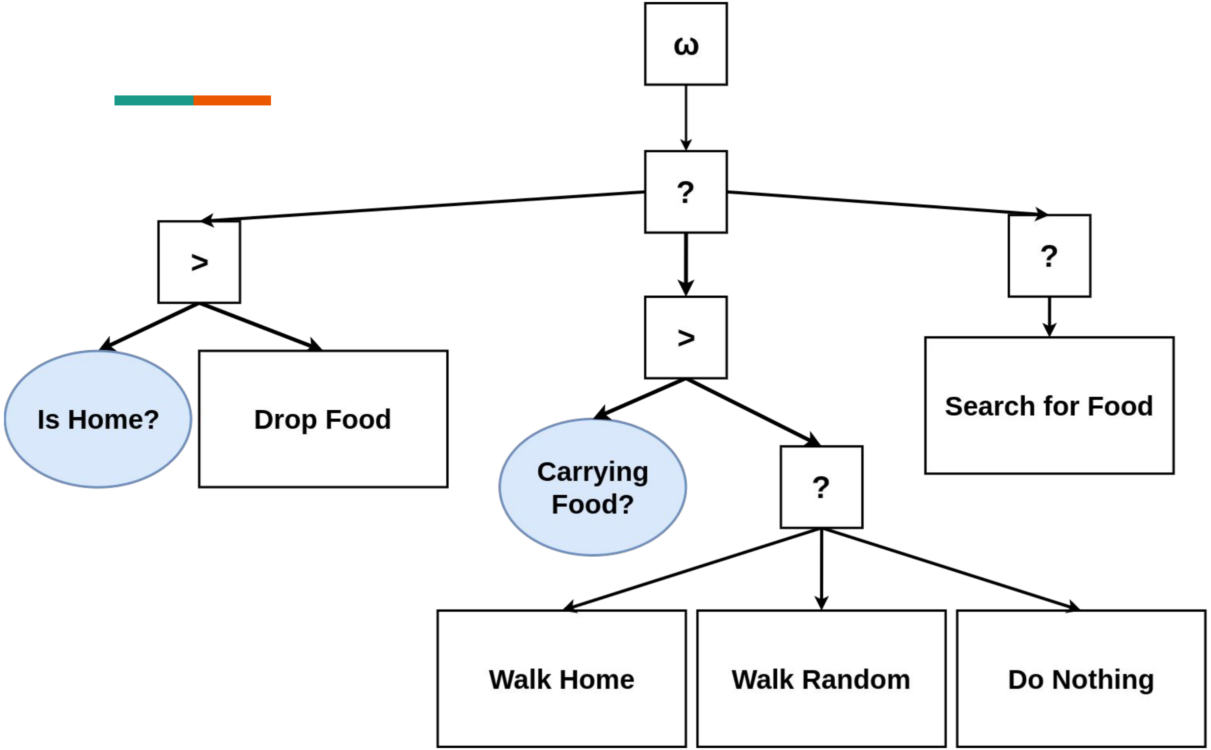
Select Nodes:

- Try each child from left to right
- Move control to the first successful child
- Return false if all children return return false
- Can be thought of as if/then/else block



Sequence Nodes

- Try each child from left to right
- Returns as soon as it hits a false child
- Can be thought of as an if/and/if/and/if statement



Blackboard Nodes:

- Come in the form of questions
- Are used to interact with the world
- Can be thought of as if statements

The blackboard:

The blackboard is how behavior trees handle memory. The blackboard is made up memory cells called blackboard values.

- Can be edited by behavior nodes
- Can be queried by blackboard nodes
- Pre-determined

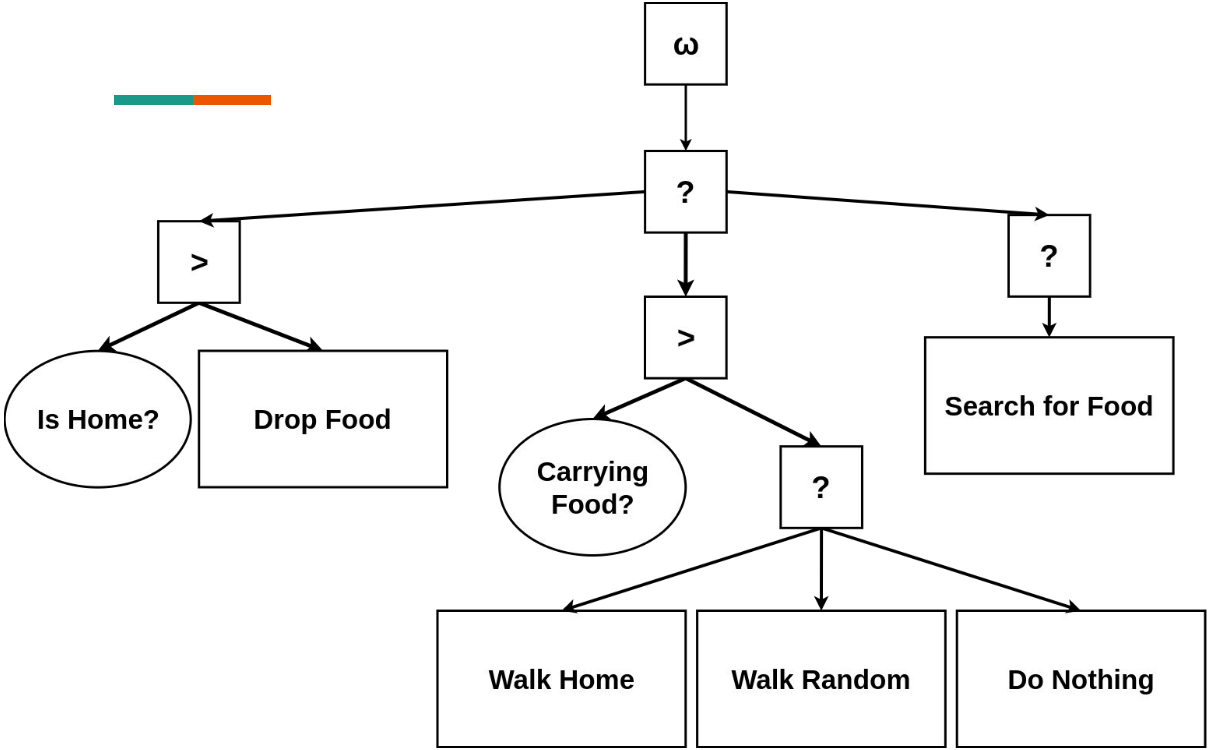
Tree traversals:

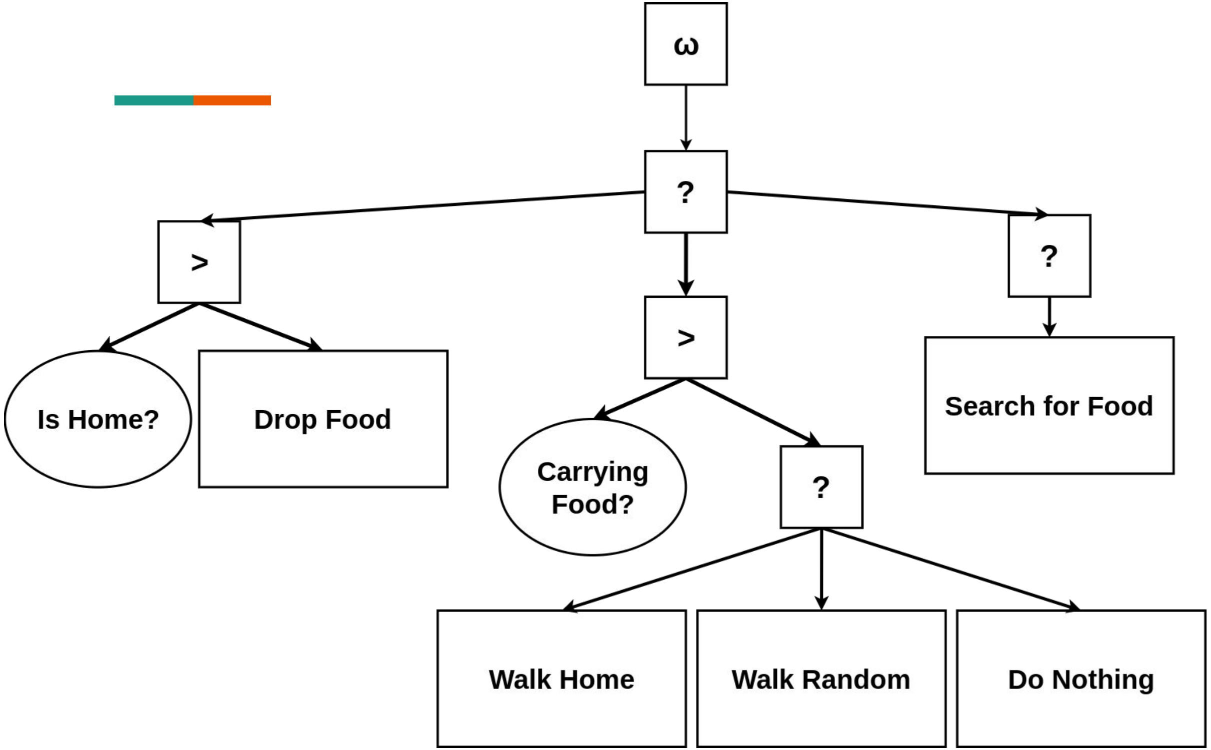
Each tick, evaluation of the BT begins at the root node, and continues recursively through the decision nodes until either a leaf is reached, or all relevant decision nodes have been queried.

- If a leaf node is reached, the leaf is queried, and the result is returned as the traversal value
- If a leaf node it not reached, the traversal will abort, and evaluation will begin again next tick

Tree states:

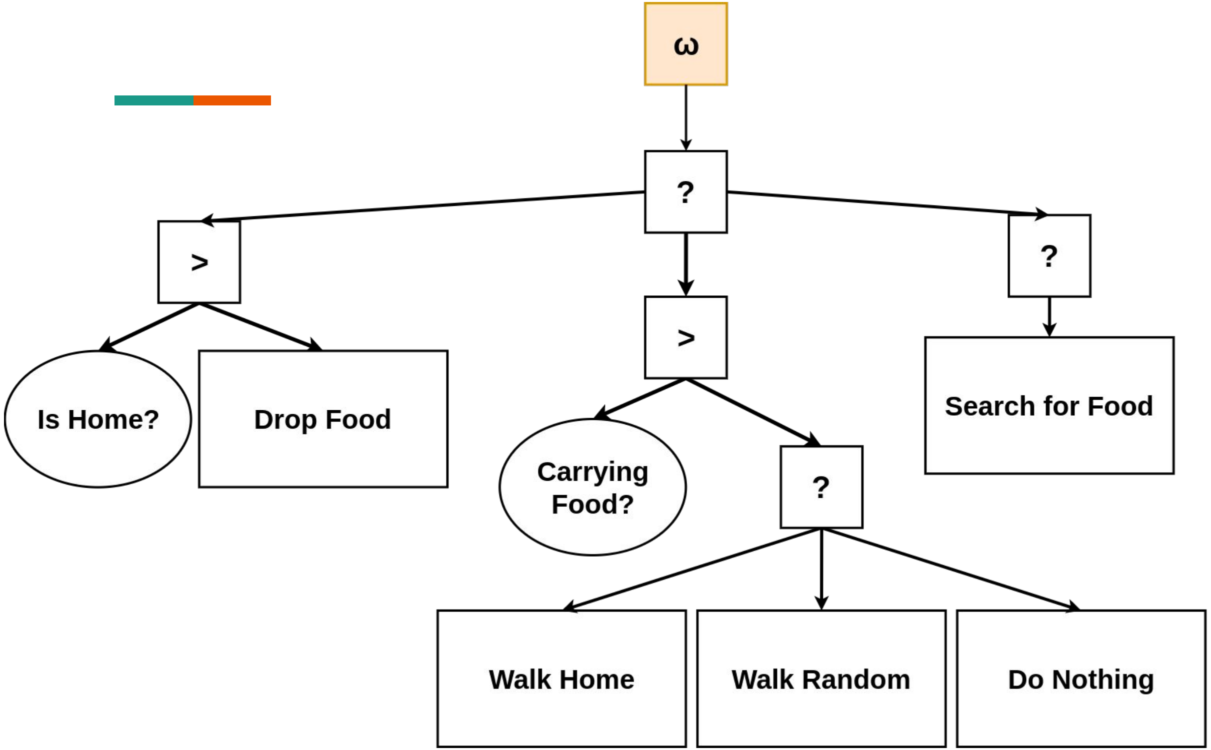
- **Running**: The behavior node will return running if it still processing its current behavior
- **Success**: The behavior node will return success, and set its internal state to running, unless it is unable to handle the behavior for some other reason
- **Failure**: Failure is a fallback state, and represents a behavior that is unable to be run at that time


ω
?
>
?
>
Is Home?
Drop Food
Search for Food
Carrying Food?
?
Walk Home
Walk Random
Do Nothing



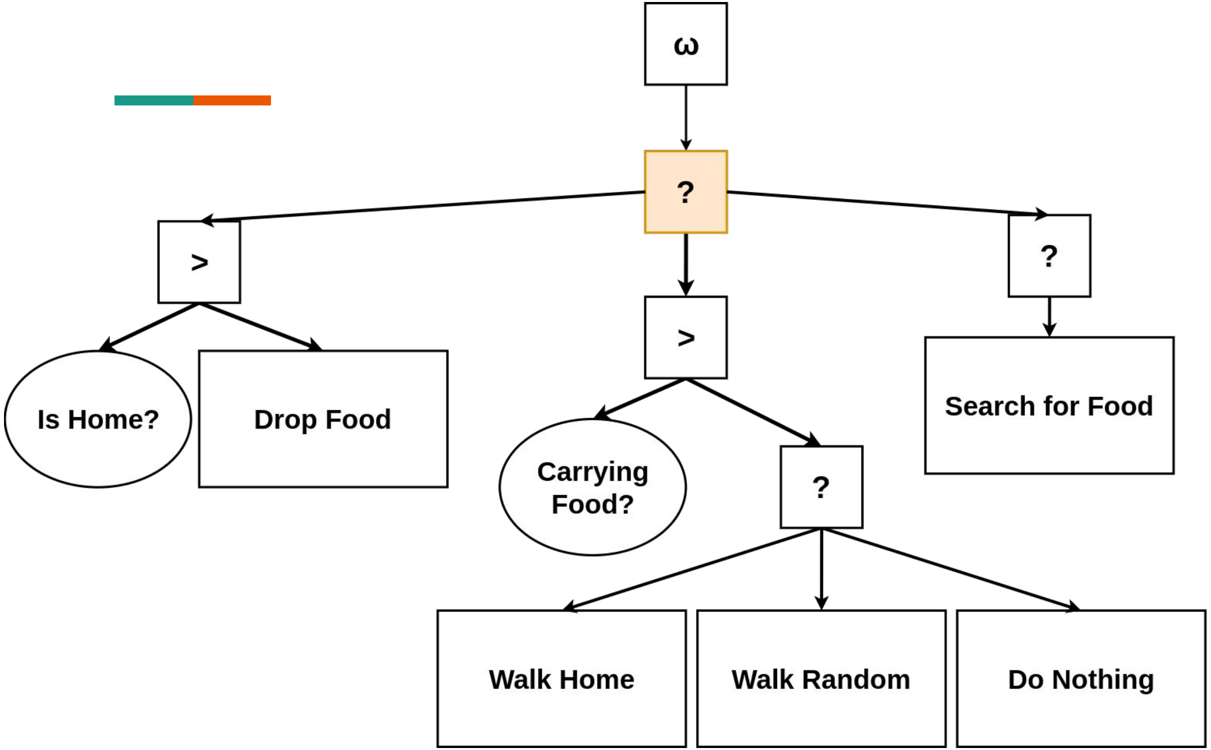
The termite blackboard:

- Not home
- Not carrying food



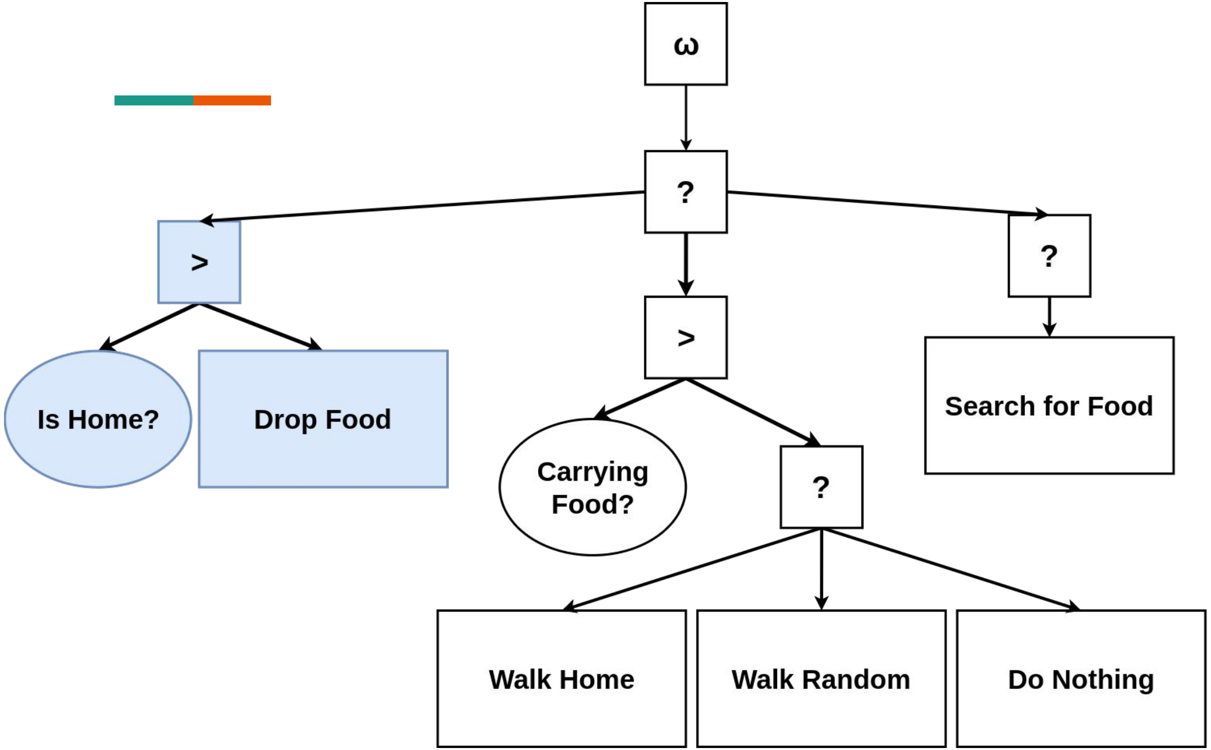
The termite blackboard:

- Not home
- Not carrying food



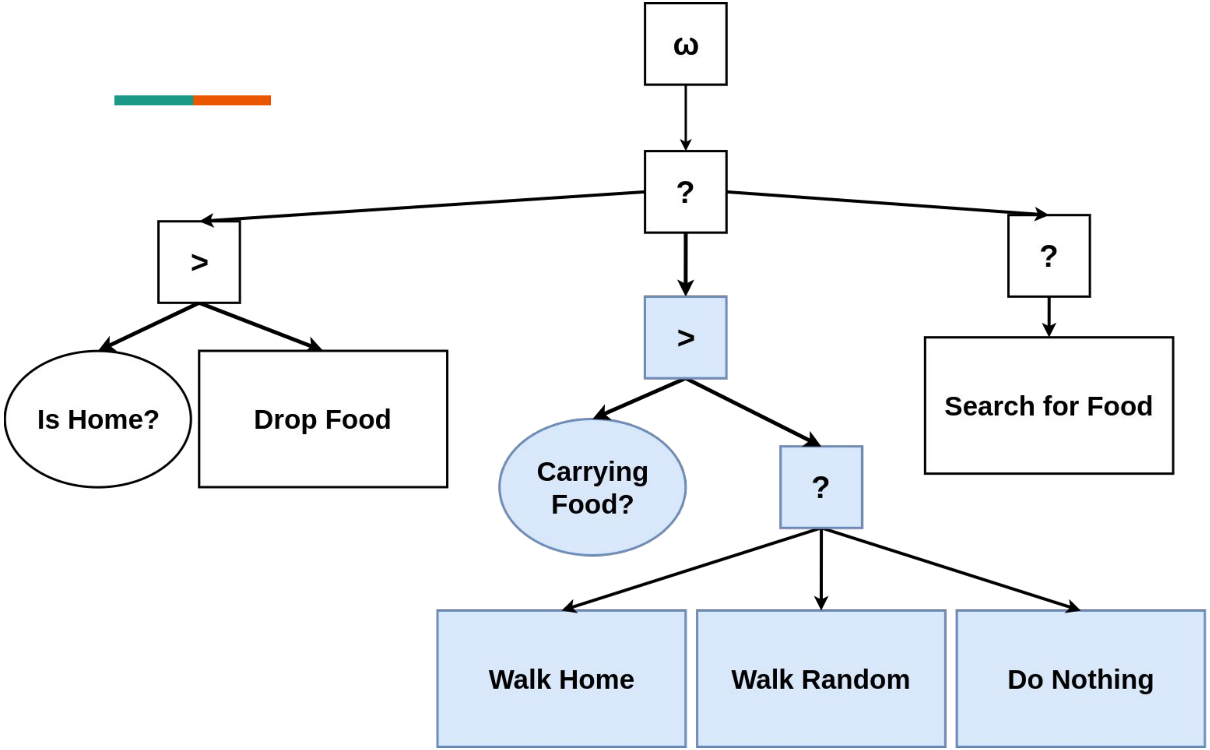
The termite blackboard:

- Not home
- Not carrying food



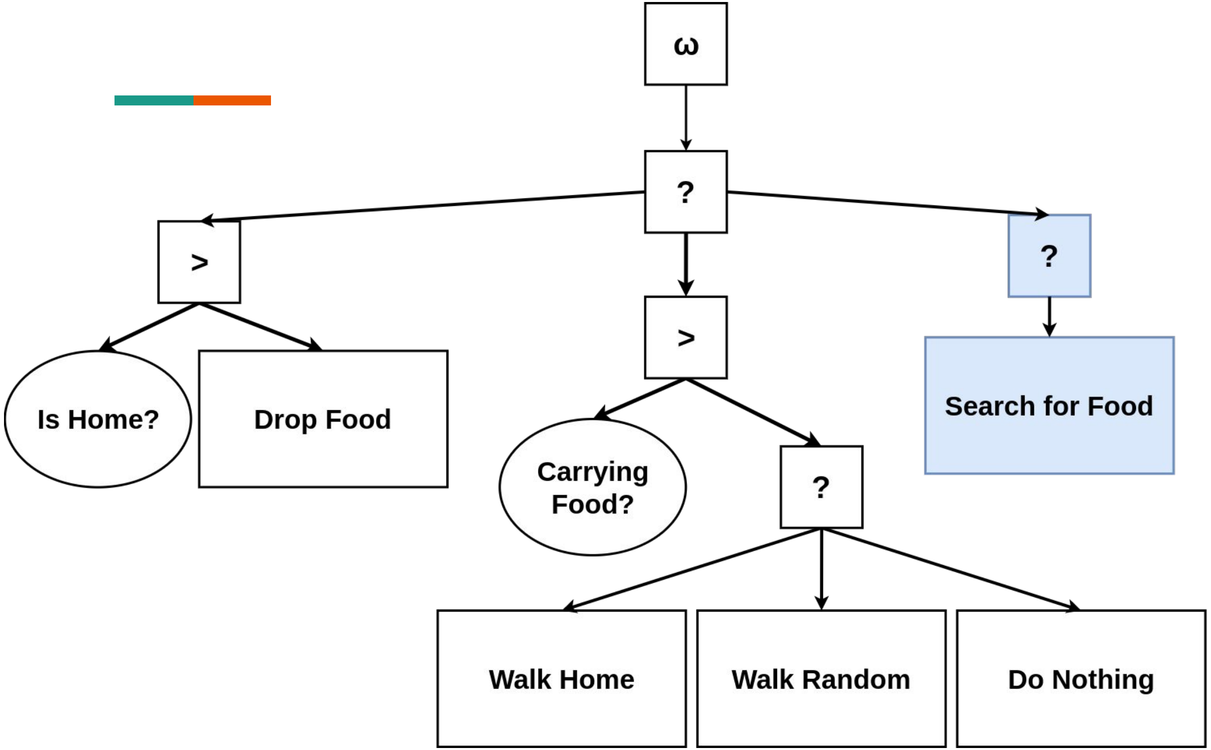
The termite blackboard:

- Not home
- Not carrying food



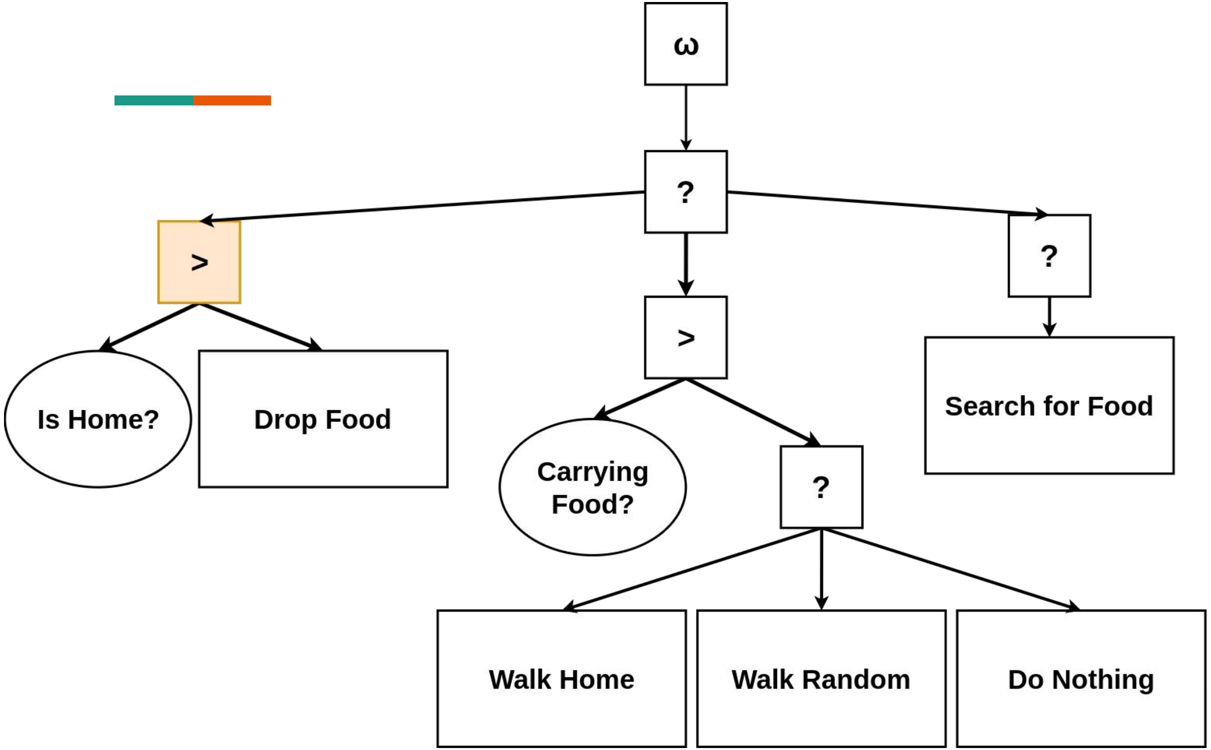
The termite blackboard:

- Not home
- Not carrying food



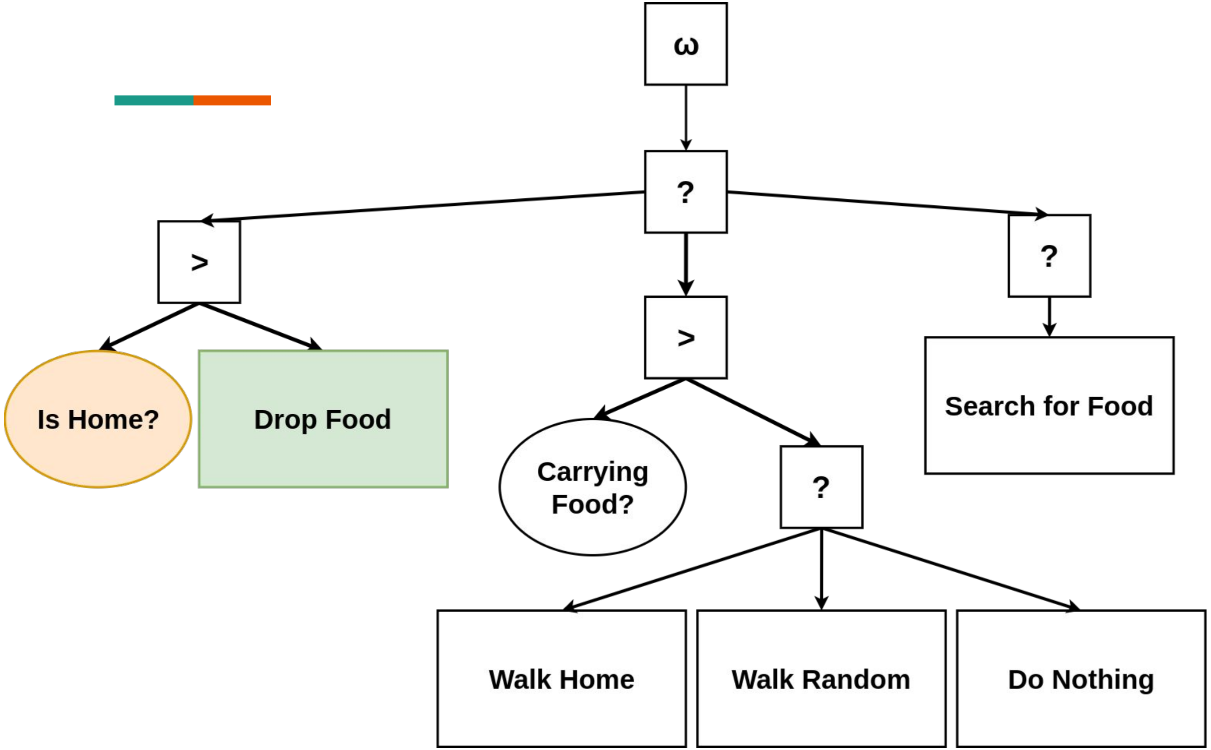
The termite blackboard:

- Not home
- Not carrying food



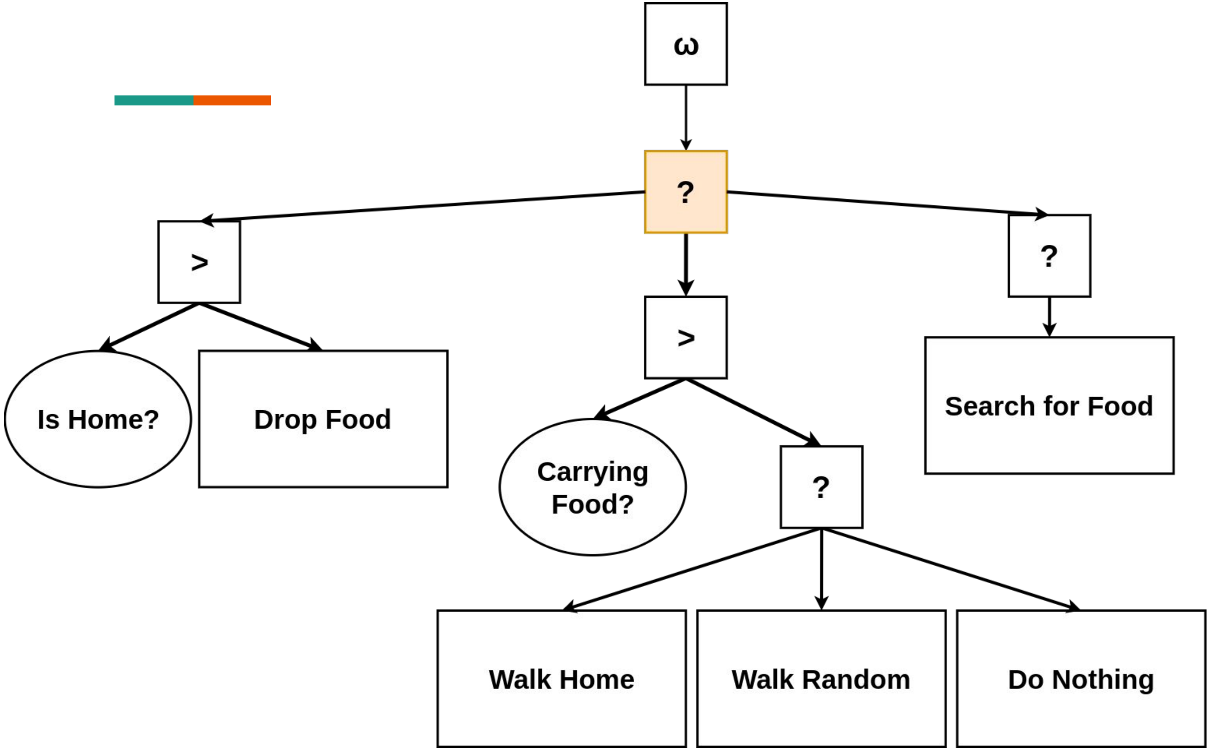
The termite blackboard:

- Not home
- Not carrying food



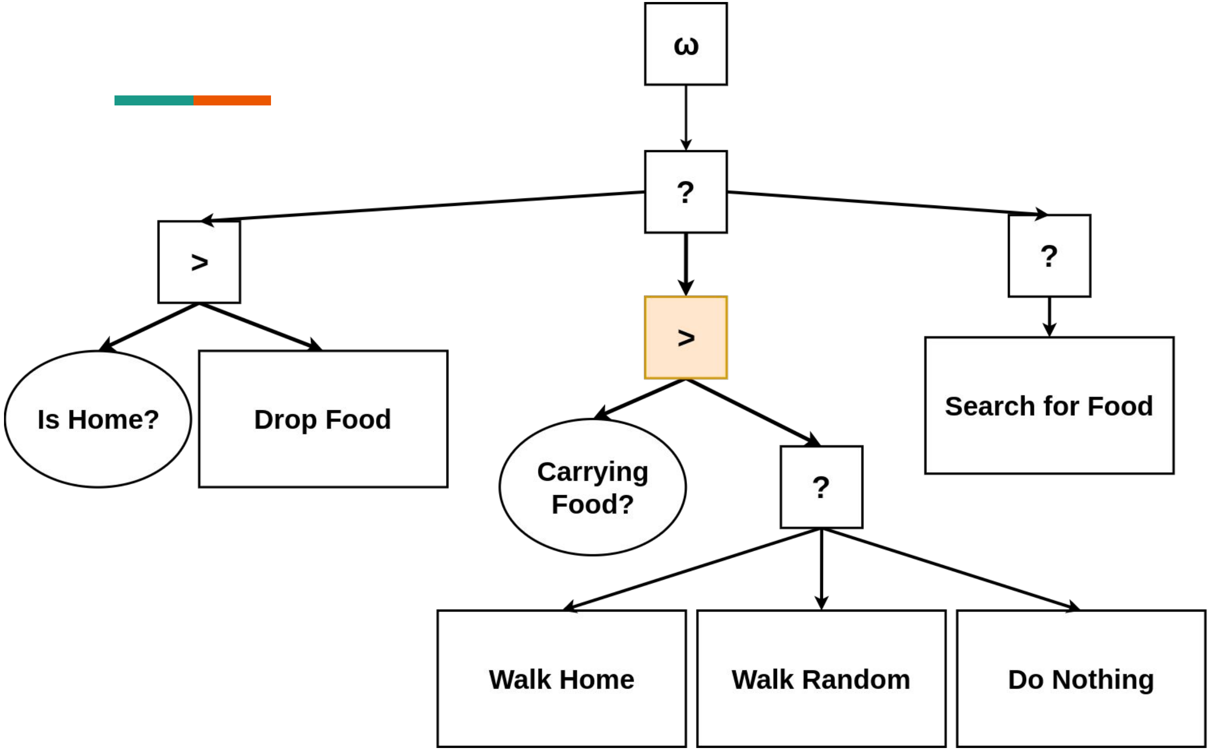
The termite blackboard:

- Not home
- Not carrying food



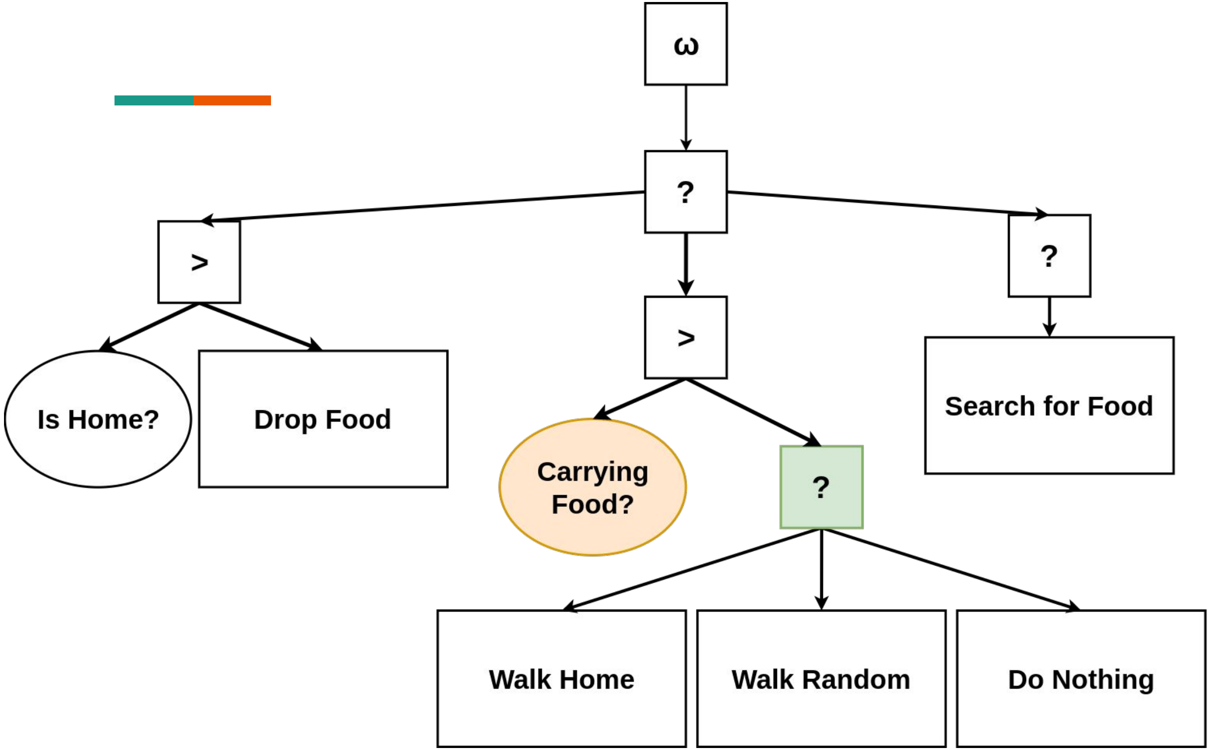
The termite blackboard:

- Not home
- Not carrying food



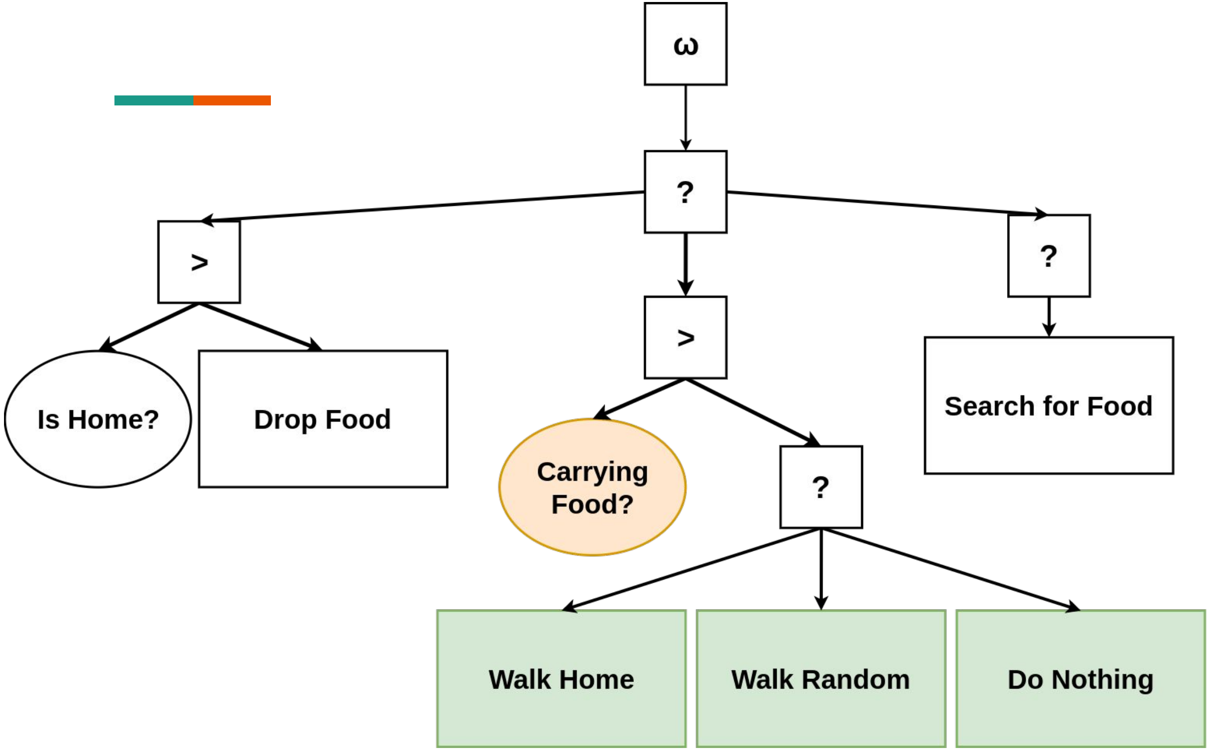
The termite blackboard:

- Not home
- Not carrying food



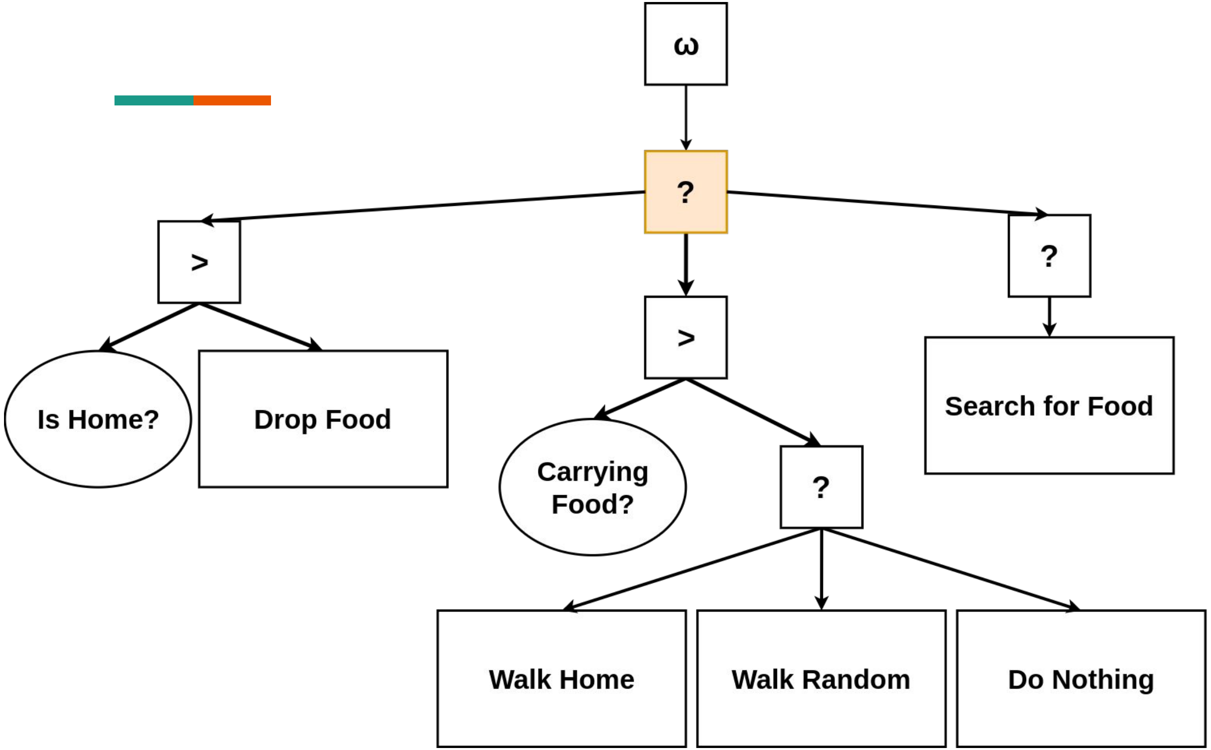
The termite blackboard:

- Not home
- Not carrying food



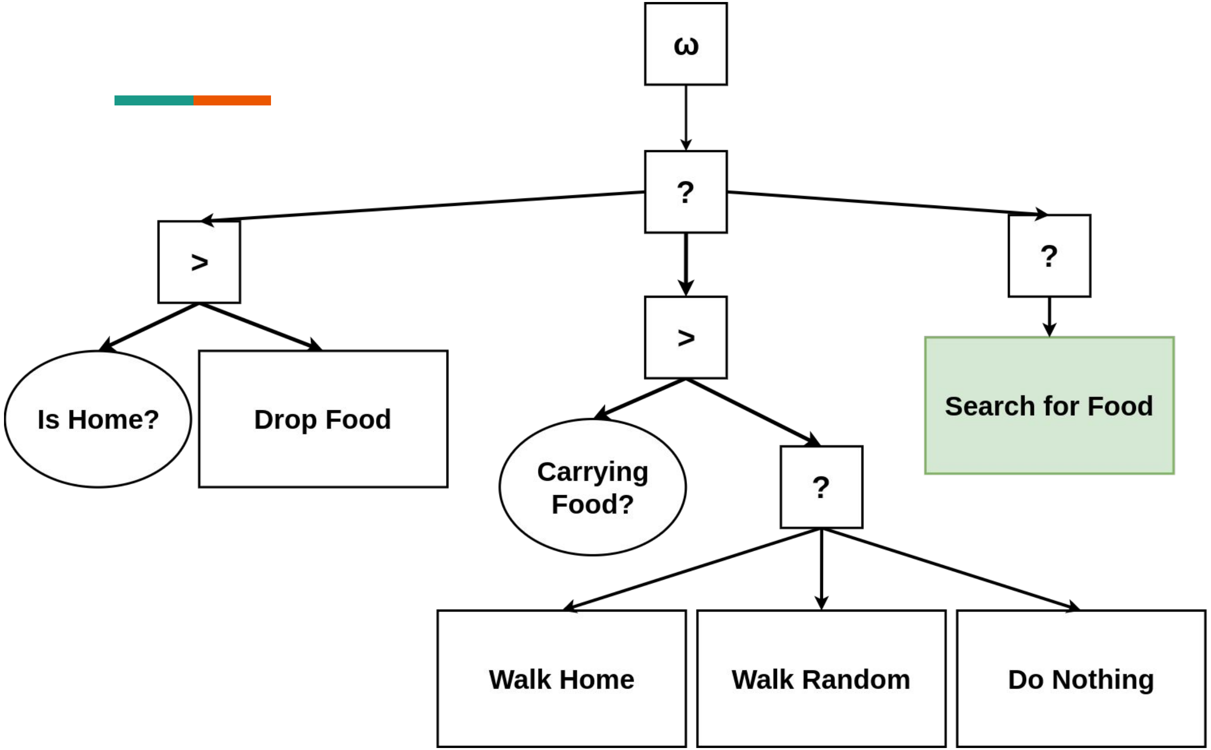
The termite blackboard:

- Not home
- Not carrying food



The termite blackboard:

- Not home
- Not carrying food



The termite blackboard:

- Not home
- Not carrying food

Genetic Programming

Genetic Programming:

- Genetic programming is a subset of artificial intelligence research that involves *evolving* programs
- Genetic programming works by taking a set of *unfit* programs (often randomly generated) and applying evolutionary pressure

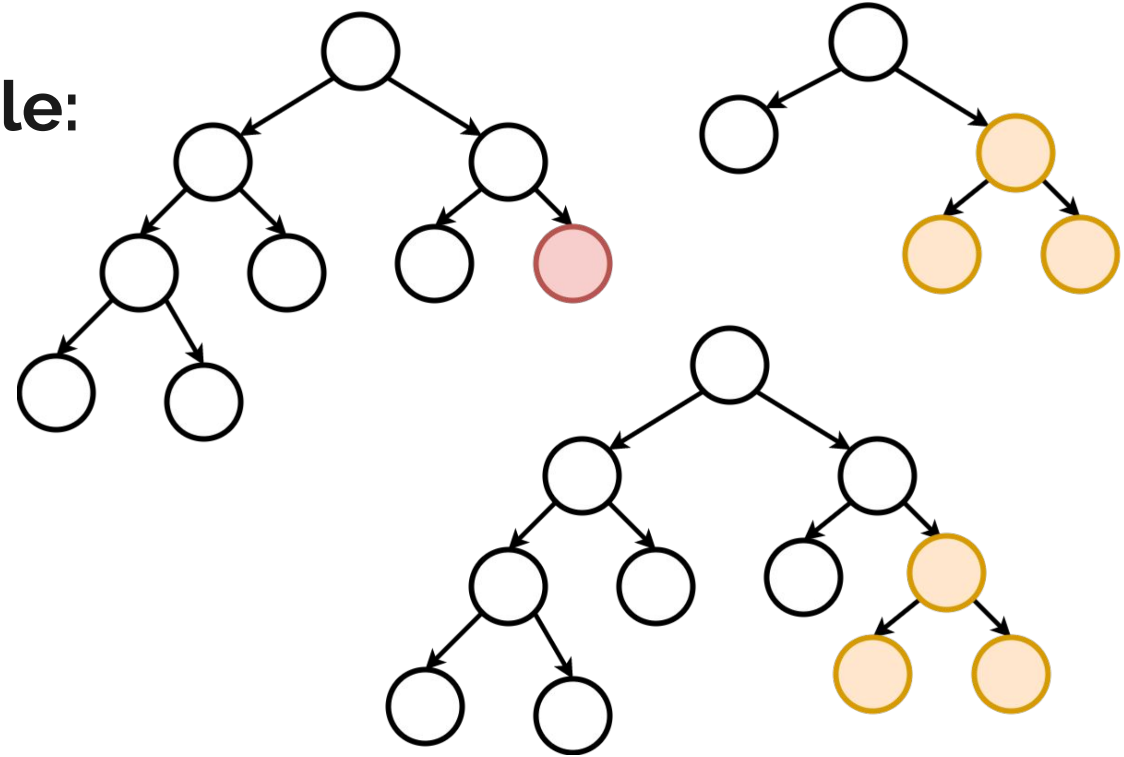
Genetic Programming & Biological Evolution:

- Programs can be thought of as organisms
- Organisms are reproduced
- There is a filter that insures the best organisms persists

Vocabulary:

- **Fitness** is the rating we give a program. It represents how well the program performed
- **Generations** are comparable to human generations
- **Crossover** is the sexual reproduction of two programs. Crossover results in new *child* programs

Crossover example:



Evolving Behavior Trees

Swarm modeling:

Swarm robotics is the field of research surrounding small, autonomous robots interacting with each other on a large scale

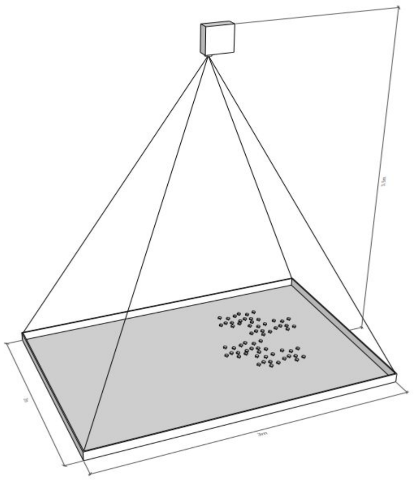
- The inspiration for swarm-robotics comes from colony-representative species such as bees, or termites
- Swarm robotics is interested in using the emergent properties of swarms to create useful robotic behaviors out of relatively simple agents
- Behavior trees can be used to model the individual agents in a swarm. These agents can then be tested or studied in a simulation environment

Why behavior trees?

"Evolved controllers are often difficult to understand, limiting our ability to predict swarm behaviour. We suggest behaviour trees are a good control architecture for swarm robotics, as they are comprehensible and promote modular reuse."

Jones et al.

Kilobots:



- A Kilobot is a small, cheap, physical robot
- Each Kilobot represents an agent in a swarm
- Each Kilobot is equipped with two vibrating motors, which allow the bot to turn and move forward
- An upwards facing photo detector for environment sensing

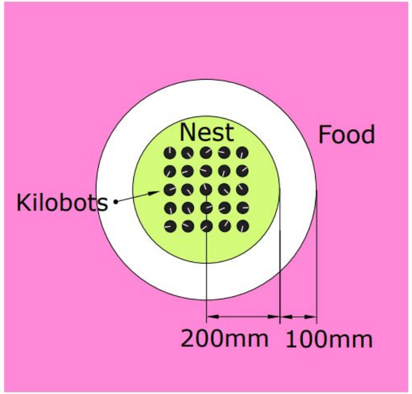
Image taken from [2]

Foraging:

A common task for modeling swarm behaviors is foraging, which is the task of moving away from a home area, collecting food and returning home.

- Includes many agent interactions
- Encourages cooperation
- Has real world applications

Foraging environment:



- At the centre of the arena is a circular nest region
- Surrounding this is a gap, then beyond that is the food region
- A kilobot which moves into the food region is regarded as having picked up an item of food, a kilobot which is carrying an item of food that enters the nest region is regarded as depositing the food in the nest

Image taken from [2]

Kilobot behavior structure:

Genetic programming works by combining a predefined set of operations in a structured way.

When evolving behavior trees, the structure is built out of behavior tree components: **decision nodes**, **behavior nodes**, and **blackboard values**. These elements are combined and trained using genetic programming.

Structure: Decision Nodes

The following decision nodes exist:

- Sequence
- Selection
- Always succeed
- Always fail
- Repeat

Structure: Behavior Nodes

The following behavior nodes exist:

- Move forward for one tick
- Turn left for one tick
- Turn right for one tick
- Always succeed
- Always fail

Structure: Blackboard

Name	**Values**:
motors	*on, off, left, right*
Δdensity	*Change in kilobot density*
$\Delta distance_{nest}$	*Change in distance to home*
$\Delta distance_{food}$	*Change in distance to food*
detected_food	*true, false*
carrying_food	*true, false*

Structure: Blackboard Queries

The Kilobot can query its blackboard values in the following ways:

- Compare two blackboard values against each other
- Compare a blackboard value against a constant

Example:

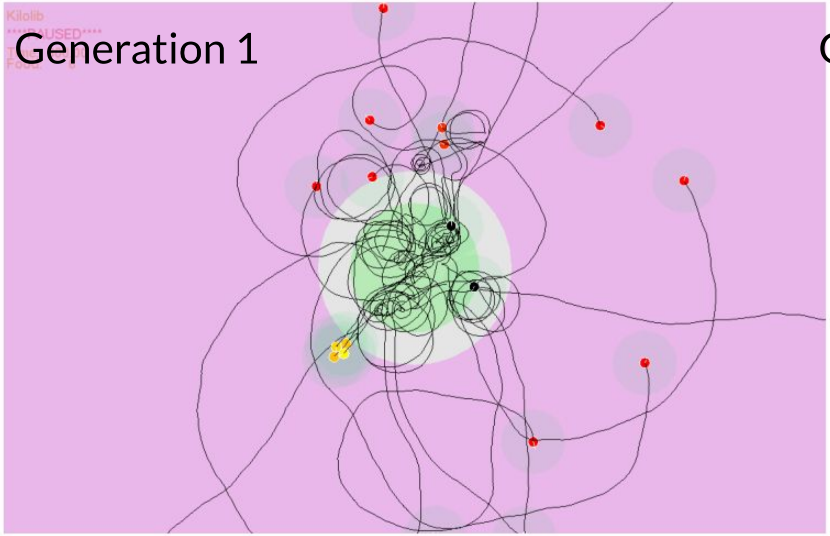
- $\Delta dist_{food} > Motors$

Evolving behavior trees

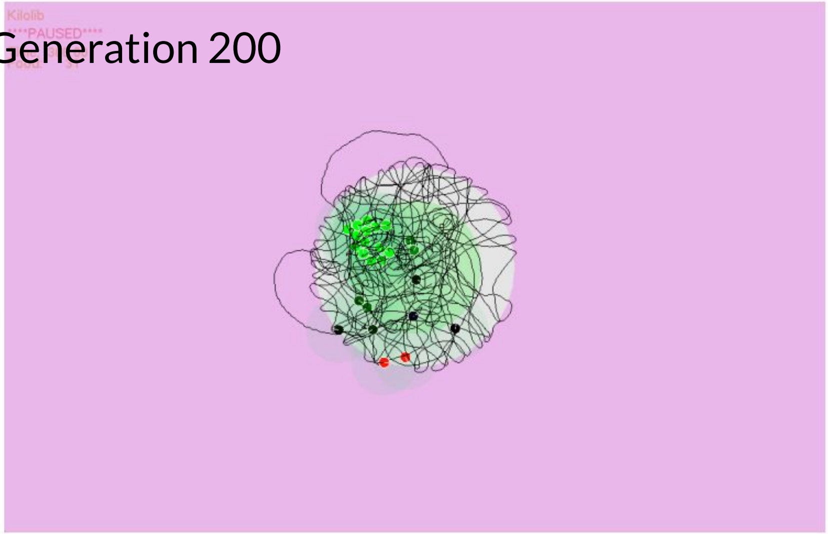
Evolution proceeds as follows:

- The population of n_{pop} is evaluated for fitness by running 10 simulations for each individual, each simulation with a different starting configuration
- The starting position is always a 5x5 grid with 50mm spacing in the centre of the nest region
- The simulation runs for 300 simulated seconds

Pathing for evolved Kilobots:



Generation 1



Generation 200

Image taken from [2]

Taken from [1]

Evolutionary discovery

- Fitness rises fast after the first generation
- The authors note that this is because Kilobots that only move forward are still collecting some food
- The best Kilobot lineage is significantly better than the rest

The best Kilobot:

- Not all of the hardwired capabilities are used
- Only *detected_food*, $\Delta dist_{food}$, and $\Delta dist_{nest}$ are used
- This is not the case for all agents: combined together, each capability was used at least once
- There is no obvious correlation between the features used and the fitness of the individual, perhaps indicating that there are multiple ways to solve this foraging problem

The best Kilobot:

- The first clause causes the kilobot to move forward as long as it is not in the food region
- If it enters the food, the second clause comes into play, performing a series of left turns and forward movements until it moves out of the food region
- Behaviour will then revert to the first clause and it will move forward again, likely hitting the nest region

```
selm3(
  seqm2(                          Move forward until in food
    ifge(0, detected_food),
    mf()),
  seqm8(                          Turn and forward until out of food
    ml(),
    ifge(Δdist_food, 0),
    mf(),
    ml(),
    mf(),
    ifge(Δdist_food, 0),
    mf(),
    mf()),
  seqm3(
    ml(),
    repeat(5,
      iflt(Δdist_nest, −0.058530)),
    ml()))
```

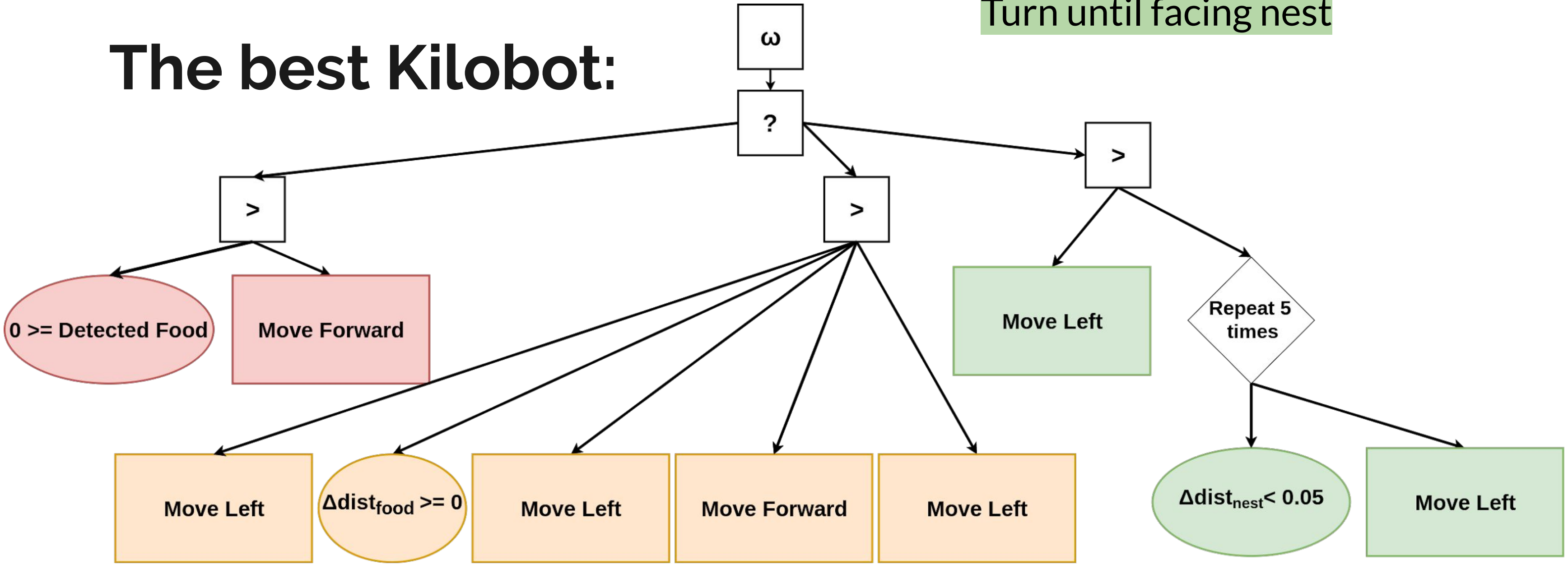
Table taken from [2]

The best Kilobot:

Move forward until in food

Turn and forward until out of food

Turn until facing nest



Results:

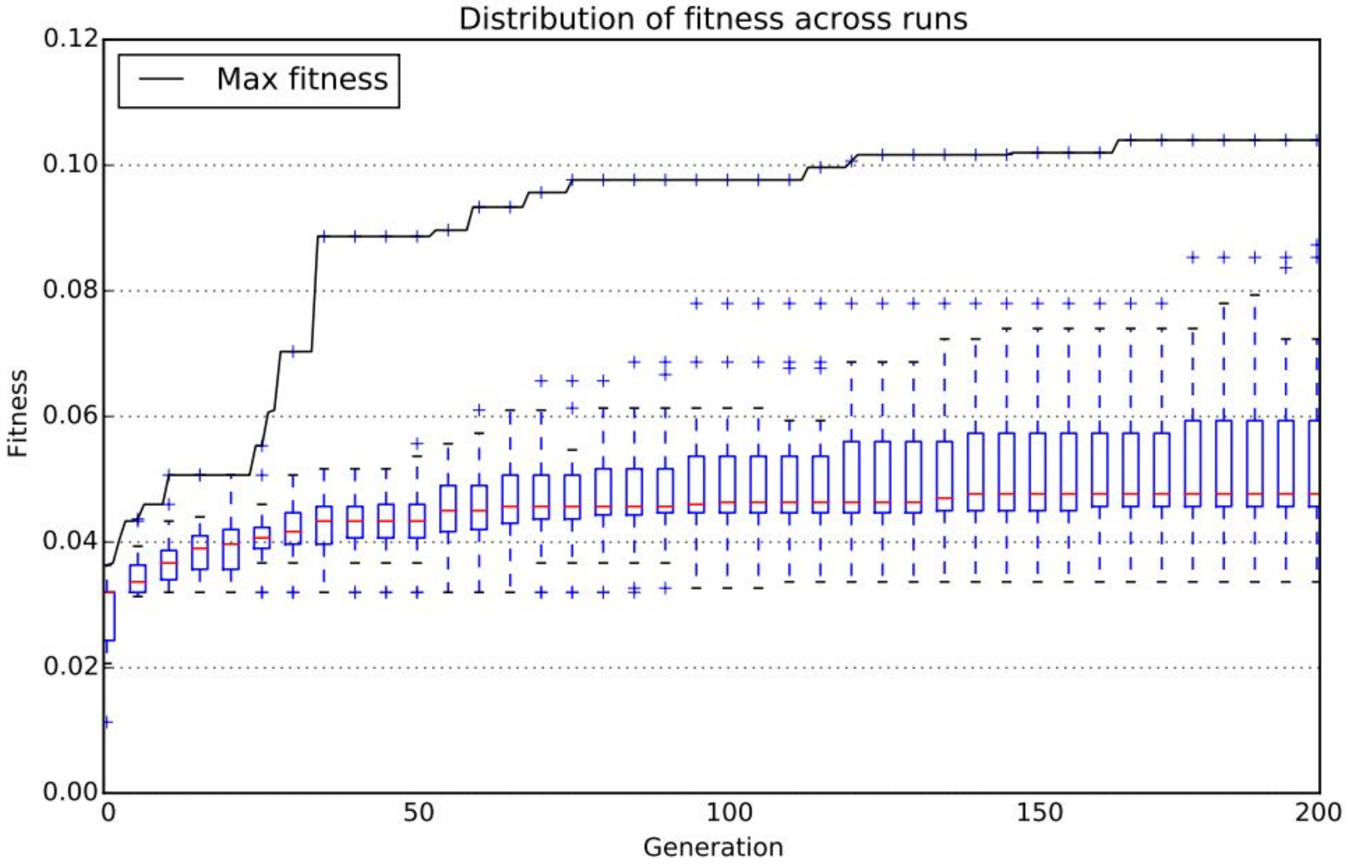


Image taken from [2]

Conclusions:

- Behavior trees are a human-readable and powerful agent-modeling solution
- Behavior trees can be evolved using genetic programming
- Evolved behaviors are viable and interesting

Questions?

Thanks to Kristin Lamberty and Nic Mcphee for making this talk possible.

Parameter	Value	Description
n_{gen}	200	Generations
t_{test}	300	Test length in seconds
n_{pop}	25	Population
n_{elite}	3	Elite
t_{size}	3	Tournament size
p_{xover}	0.8	Crossover probability
p_{mutu}	0.05	Probability of subtree replacement
p_{muts}	0.1	Probability of subtree shrink
p_{mutn}	0.5	Probability of node replacement
p_{mute}	0.5	Probability of ephemeral constant replacement

Table 3: Parameters for a single evolutionary run

Citations:

[1] M. Colledanchise and P. Ögren. Behavior trees in robotics and AI: an introduction. *CoRR*, abs/1709.00084, 2017.

[2] S. Jones, M. Studley, S. Hauert, and A. Winfield. Evolving behaviour trees for swarm robotics. In *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics, pages 487–501. Springer, 3 2018.

[3] Wikipedia. Behavior tree — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Behavior%20treeoldid=922038836, 2019. [Online; accessed 15-November-2019].

[4] Wikipedia. Genetic programming — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Genetic%20programmingoldid=921953038, 2019. [Online; accessed 15-November-2019].

[5] Wikipedia. Swarm robotics — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Swarm%20robotics&oldid=921430950, 2019. [Online; accessed 03-November-2019].