Robotic Programming Environments - Behavior Trees and Finite State Machines in robotics (winter 2023/2024)

MSc Łukasz Janiec

Wrocław University of Science and Technology
Faculty of Electronics, Photonics and Microsystems
Department of Cybernetics and Robotics

December 11, 2023

Wrocław University of Science and Technology

HR EXCELLENCE IN RESEARCH

unite! University Network for Innovation, Technology and Engineering

Outline of the presentation

History of Behavior Trees

Behavior Trees (BTs) have their origins in the field of artificial intelligence and robotics.
The concept of Behavior Trees was introduced by Michaele Colledanchise and Petter Ogren in their paper titled "Behavior Trees in Robotics and AI," which was published in 2009.
The authors aimed to provide a formalism for describing the control flow of robotic systems in a way that is both intuitive and modular.

Fundamentals of Behavior Trees

Behavior Trees are hierarchical structures used for defining the decision-making process in autonomous systems, such as robots and game characters. The tree structure is composed of nodes, each representing a specific behavior or task.

BT nodes categorization

Composite Nodes:

- **Sequence**: Executes child nodes in sequence until one fails.
- **Selector**: Executes child nodes in sequence until one succeeds.
- **Parallel**: Executes child nodes concurrently.

Decorator Nodes:

- Modify the behavior of child nodes. Examples include conditions or time limits.

Leaf Nodes:

- Represent basic behaviors or tasks that cannot be further subdivided.

The tree structure allows for a clear representation of the system's decision-making process, providing modularity and ease of understanding.

Suggested libraries

- `https://py-trees.readthedocs.io/`
- `https://www.behaviortree.dev/`

History of FSM

Finite State Machines (FSMs) have a long history in computer science and engineering. The concept of finite automata, from which FSMs evolved, dates back to the early 20th century.
However, the formalization and popularization of FSMs in the context of digital systems and automation began in the mid-20th century.
In the 1940s and 1950s, mathematicians and engineers, including John von Neumann and Claude Shannon, laid the theoretical foundations for digital systems and automata, contributing to the development of FSMs. The practical application of FSMs in control systems and automation gained momentum in the following decades.

Fundamentals of FSM

A Finite State Machine (FSM) is a mathematical model consisting of a set of states, transitions between these states, and actions associated with each transition. It operates in discrete steps, moving from one state to another based on input conditions.

The key components of an FSM

- **States**: Represent distinct conditions or modes that the system can be in.
- **Transitions**: Define the conditions under which the system moves from one state to another.
- **Actions**: Associated with transitions, actions are tasks or behaviors performed when transitioning between states.
- **Inputs**: External stimuli or events that trigger state transitions.

FSMs can be categorized into two main types: deterministic FSMs, where the next state is uniquely determined by the current state and input, and non-deterministic FSMs, where multiple transitions are possible for a given state and input.

Suggested libraries

- `https://python-statemachine.readthedocs.io/`

Extra task for up to 14 points (1/2)

Using py_trees, implement example simple BT control of the autonomous mobile robot. Examples - choose one of these or think up your own.

Use of the BT in robotics (7 pts)

- Design a BT to control a simulated robot - it should include behaviors for moving forward, avoiding obstacles, and reaching a goal. Use a Selector node to prioritize behaviors based on the context.
- Design a BT that utilizes parallel nodes to integrate sensory information. For instance, implement a robot that can perform both exploration and obstacle avoidance simultaneously using parallel tasks.
- Develop a BT to represent a sequential task execution. For example, create a tree that guides a robot through a series of waypoints, ensuring that each waypoint is reached before moving on to the next.

Extra task for up to 14 points (2/2)

Using `python-statemachine`, implement example FSM for control of the autonomous mobile robot. Examples - choose one of these or think up your own.

Use of the FSM in robotics (7 pts)

- Implement a Finite State Machine to control a robot's navigation behavior. Define states such as `Explore`, `AvoidObstacle`, and `ReachGoal`. Transitions between states should be triggered by specific conditions (e.g., obstacle detected, goal reached).
- Implement an FSM that responds to external events. For instance, create states for `Idle`, `ProcessingData`, and `Error`, where transitions are triggered by events such as data arrival or an error condition.
- Design an FSM for a robot that interacts with humans. Define states like `WaitForCommand`, `ExecutingCommand`, and `EngageHuman`. The transitions can be triggered by voice commands or gestures (events).

Simplifying assumptions

- You can get partial credit for only designing correct BT and FSM - use draw.io or similar software (a plan on a sheet of paper is fine too).
- You should implement ”stubs” instead of the full functionality for BT nodes and states/transitions in the FSM.
- Base as much as you can on the examples from the documentation.
- Use the integrated tools to get dot/svg/png files of the BT and FSM to show me your work.

Questions?

Feel free to ask questions.