Decision Making with ROS

1 March, 2017

Janyl Jumadinova
Decision Making in ROS

http://wiki.ros.org/decision_making

- The goal of this package is to implement light-weight, generic and extendable tools for writing, executing, debugging and monitoring decision making models through ROS standard tools.
- Decision making package is being actively developed by Cogniteam.
  - For single robot: it is a public package.
  - For multi robot: needs a commercial license.
The decision making system supports different types of models:

- FSM – Finite State Machines
- HSM – Hierarchical FSM
- Behavior Trees
- CogniTAO – implementation of BDI (belief-desire-intention) architecture
A Finite State Machine, or FSM, is an abstraction used to model a device that can be in any one of a fixed number of states – each capturing some state of the system, drawn as circles, one being the start state shown with a pointed arrow.
A Finite State Machine, or FSM, is an abstraction used to model a device that can be in any one of a fixed number of states – each capturing some state of the system, drawn as circles, one being the start state shown with a pointed arrow.

While in each state, the FSM produces one or more outputs.
A Finite State Machine, or FSM, is an abstraction used to model a device that can be in any one of a fixed number of states – each capturing some state of the system, drawn as circles, one being the start state shown with a pointed arrow.

While in each state, the FSM produces one or more outputs.

Some event causes a transition from state to state – an arrow between two states, labelled with the input which activates it and the output which results.
FSM
HSM

- FSM designed in a hierarchical structure to avoid large number of states/events.
HSM

- FSM designed in a hierarchical structure to avoid large number of states/events.
- High-level states → sub-states within each high-level state → behaviors

Example:
- High-level states: Wander, Attack, Chase, and Spawn
- Chase: chase on foot, chase while riding a unicyle, chase while flying a hovercraft
HSM

- FSM designed in a hierarchical structure to avoid large number of states/events.
- High-level states → sub-states within each high-level state → behaviors
- Example:
  high-level states: Wander, Attack, Chase, and Spawn
  *Chase*: chase on foot, chase while riding a unicycle, chase while flying a hovercraft
Behavior Trees

- More general than FSM, uses hierarchical structure
- First used in Halo 2 (than Spore)
- Example:
  - Guard dog
  - Behavior defined hierarchically
  - AI system interacts with the game state at the leaves of the tree
  - Leaves allow to gather information through *conditions* (e.g., dog’s state: hungry or injured) and affect the progress through *actions*
Behavior Trees
CogniTAO (Think As One) is an implementation of the BDI architecture for both single robot missions and for multiple robots working in teams.

Main features:
- Simulate entities that can execute complex missions in dynamic environments, where it is impossible to foresee all possible decisions.
- Coupling between the decision-making and the world modeling components.
- Mixing goal-oriented and reactive control, according to the principals of BDI.
Plan

Defines the current Task to be performed, contains Start and Stop conditions, and is coupled with corresponding Plans through Allocation and Next protocols.
- A TAO is a level of a number of *Plans*.
- Each TAO defines its starting plan via the **TAO.START.PLAN** tag.
A TAO is a level of a number of Plans.
Each TAO defines its starting plan via the TAO_START_PLAN tag.
The Next Protocol takes place when a current plan ends, and essentially chooses one next plan to be performed.
▶ A TAO is a level of a number of *Plans*.
▶ Each TAO defines its starting plan via the TAO\_START\_PLAN tag.
▶ The Next Protocol takes place when a current plan ends, and essentially chooses one next plan to be performed.
▶ The Allocation Protocol takes place in a running plan, and essentially chooses (or divides) a sub-plan that will be performed.