# **行为树（Behavior Tree）**

## **前言：**

随着使用编程技术来解决越来越复杂的现实问题，传统的算术运算和逻辑已经越来越难以满足抽象和复杂的现实需求。更具体的说，抽象已经使我们从低级执行器控制和基本逻辑运算转向对更高级别概念行为的推理。有限状态机（ FSM ）和 行为树（ BT ） 是今天最常看到的两个抽象概念模型。

行为树是游戏和机器人行业经常使用的决策引擎。

## **什么是行为树？**

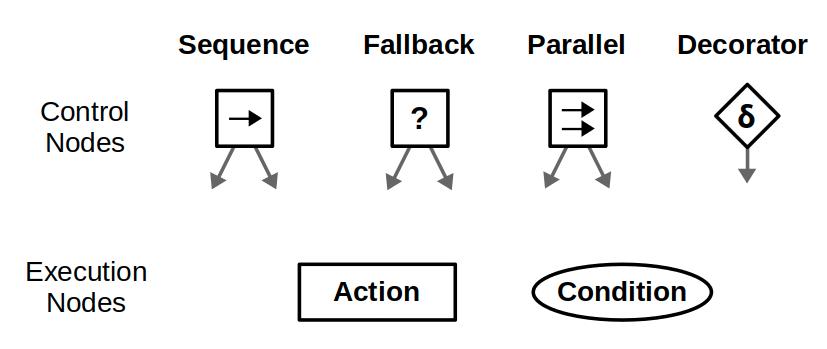
正如前言所说，我们希望使用抽象模型来帮助我们设计智能agent的复杂的行为，来简化我们的开发难度，提高我们的开发效率。一般来说，这些复杂的行为可以分解为一系列实体，这些实体可以映射到我们系统中特定的行为或者操作模式，例如 “行人开始移动”，“ 车辆变道cut in”等。每个实体都会遵循一定的规则，这些规则描述了智能agent何时按照何种规则执行这些行为，更重要的是，智能agent应该如何在这些实体中切换运行。

行为树就是这样一种抽象模型，行为树具有以下特征：

1. **行为树本质上就是树** ：它从根节点开始，被设计为以特定顺序遍历，直到达到最终状态（成功或失败）。
2. **叶子节点是可执行的行为：**每个叶子都会做一些事情，无论是简单的检查还是复杂的操作，并且会输出一个状态（成功、失败或运行）。
3. **内部节点控制树遍历：**树的内部（非叶）节点将接受其子节点的结果状态，并应用自己的规则来指示接下来应扩展哪个节点。

## **行为树术语**

如下图所示，展示了行为树不同的节点分类：



**节点分类：**

控制节点（内部节点：sequence，fallback，parallel，decorator）

执行节点（叶子节点：操作节点、条件节点）

行为树的执行本质上是采用离散的单步更新的方式，我们把BT更新一次称为一次 **tick，** 行为树的遍历或多次执行可以成为**ticks** 。

BT通常以某个指定的速率开始tick，其子节点会根据树的构造方式递归地tick。节点被执行tick后，它会向其父节点返回**状态**，可以是***Success***、***Failure***或***Running***。

**执行节点**是 BT 的叶子节点，可以是执行动作（**Action） 节点**，也可以是条件（**Condition）**节点。唯一的区别是**Condition**节点只能在单个tick内返回*成功*或*失败，而***Action***节点可以跨越多个tick并且可以返回Running*直到它们到达最终状态。一般来说，**Condition**节点代表简单的检查（例如，“车辆是否到达指定地点”），而**Action**节点代表复杂的动作（例如，“导航到指定地点”）。

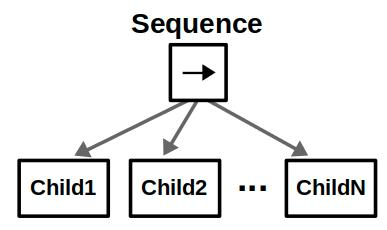
**控制节点**是内部节点，定义了如何根据其子节点的状态遍历 BT。重要的是，控制节点的子节点可以是执行节点或控制节点本身。**Sequence**、**Fallback**和**Parallel**节点可以有任意数量的子节点，但它们处理所述子节点的方式有所不同。**装饰器**节点必须有一个子节点，并使用某种自定义策略修改其行为。

### **Composites**



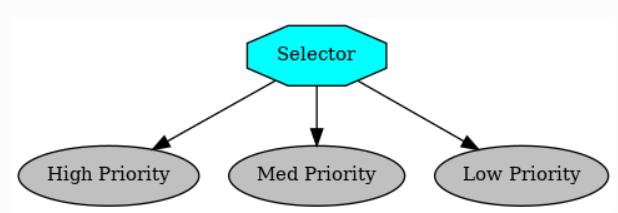
行为树的复合类型，一个Composites可以添加多个控制子节点。Composites负责在一个tick中，通过行为树的路径决定各个控制节点的执行顺序。Composites节点是所有控制节点的父类。其中的Selector是决策类，Sequence和Parallel属于工厂类

### **Sequence nodes**



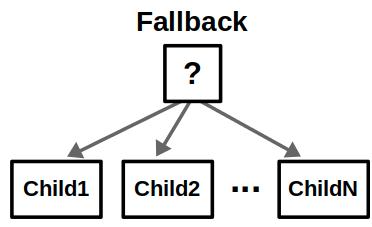
序列节点按顺序执行子节点，直到一个子节点返回Failure或所有子节点返回success.

### Selector nodes



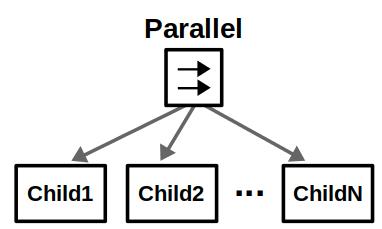
选择器节点根据级联优先级执行子任务，直到一个子节点返回success或者所有节点返回Failure.

### **Fallback nodes**



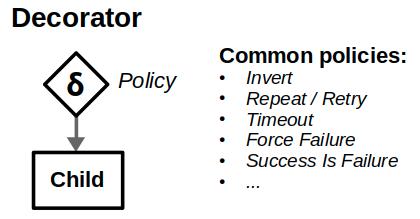
回调节点按顺序执行子节点，直到一个子节点返回success或所有子节点返回Failure。

### **Parallel nodes**



并行节点将“并行”执行其所有子节点。当至少 M 个子节点（1 到 N 之间）成功时，并行节点返回 Success；当所有子节点都失败时，并行节点返回 Failure。

### **Decorator nodes**



装饰器节点使用自定义策略修改单个子节点。装饰器有自己的一套规则来更改“装饰节点”的状态。例如，“Invert”装饰器会将成功更改为失败，反之亦然。虽然装饰器可以为您的行为树库增加灵活性，但您应该尽可能坚持使用标准控制节点和通用装饰器，以便其他人可以轻松理解您的设计

## 行为树软件库

目前主流的行为树软件库有 [py\_trees](https://py-trees.readthedocs.io/en/devel/index.html)和 [BehaviourTree.CPP](https://www.behaviortree.dev/)