How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees

Michele Colledanchise, Student Member, IEEE, and Petter Ögren, Member, IEEE

摘要

行为树是一种组织混合动态系统交换结构的方法，最初引入计算机游戏编程社区。在本文中，我们分析了行为树表示如何提高混合动态系统的模块性，以及在将两个行为树合并为一个更大的行为树的情况下，如何在这些系统的组合中保留关键的系统属性。我们还展示了如何将行为树模块视为顺序行为组合、包容架构和决策树的概括。这三种工具功能强大，但差异很大，它们在行为树组合中以自然的方式统一可能是它们在游戏社区中受欢迎的原因。最后，我们通过一组示例说明了所提出的分析工具如何应用于机器人控制行为树模块。

关键词：行为树；决策树；有限状态机；混合动力系统；模块化；包容体系结构；顺序行为组合

1 引言

行为树是一种组织混合动力系统切换结构的方法，它最初是在计算机游戏行业中开发的，作为增加游戏内对手控制结构的模块化的工具。在这个价值数十亿美元的行业中，模块化是实现代码可重用性、功能增量设计和功能高效测试的关键属性。随着业界的发展，行为树也开始受到学术界的关注。行为树也被广泛用于机器人操作，模块化是使用行为树的关键原因，主要的优点是，单个行为可以很容易地在另一个更高级别行为的上下文中重用，而不需要指定它们如何与后续行为相关。当从较小的模块组成更大的系统时，关键的系统属性可以被保留时，模块化的好处就更加明显了。我所在的课题组中研究的课题是构建高效迅速的系统平台统筹多机器人智能化感知场景，多机器人控制等，实现灾难场景下的智能化感知、决策、控制。目前阶段，计划添加行为树模块，用于机器人任务的生成及执行过程，来辅助机器人面对突发事件的处理，提高执行效率。[1]

2 相关工作

行为树起源于计算机游戏行业，主要用来对游戏的非玩家角色的行为进行建模。行为树在当今的游戏领域应用广泛，包括在实时战略游戏、第一人称射击游戏、平台游戏以及大型地剧情交互式游戏等有着出色表现。有限状态机一直作为机器人的主要控制体系架构活跃在机器人领域。行为树的出现后逐渐取代了其在机器人领域的位置。自从将行为树引用的机器人控制体系中，不管是工业生产中的机械臂、地面机器人、还是无人机以及水下机器人都在逐渐采用行为树作为机器人的控制结构。

如今行为树的创建分为两种，一种根据具体的场景手工设计行为树作为整个系统的控制部分，另一种通过学习或者规划方法自动生成所需要的行为树。这两种方法有利有弊，在手工设计方面，主要是以人类的思维去考虑整个系统的运行逻辑然后手动设计行为树结构，但遇到较为庞大系统时，就带

来了巨大的工作量。在自动生成行为树方面，自动生成的行为树通过给定的条件结合智能算法自动构建任务所需要的行为树，虽然省去了手动设计，但容易出现以人类思维难以理解的树结构，带来了行为树的验证与可读性问题。目前使用较多的还是手工设计，但是将行为树与智能算法的结合也在逐渐发展，两者的有效结合才能设计出完善的行为树结构。

国外游戏公司将行为树应用到游戏人工智能，展现出了不错的效果，国内大型游戏公司受其影响下，也在逐渐将行为树融入到游戏设计中。腾讯公司作为国内游戏行业的代表，在 2016 年发布基于行为树的开源项目“Bebaviac”。Behaviac 是以行为树作为基础框架的游戏 AI 的开发组件，也是设计游戏控制结构的工具，并目该项目支持行为树、有限状态机、层次任务网络等控制结构。目前公司利用 Behaviac 组件进行大量的游戏开发，较为成功的应用案例包括王者荣耀，QQ 飞车等热门游戏。

行为树在国外研究主要集中在游戏领域，机器人领域方面以及结合智能算法设计行为树，国内的研究在游戏领域以及将行为树作为仿真决策系统开展。[1]。

3 本文方法

文章中以教科书和游戏 AI 论文中的经典方式描述行为树。令行为树是一棵有向树，具有节点、边、根、叶、子对象和父对象的通常定义。在行为树中，每个节点都属于图 1中列出的五个类别之一。叶节点是动作或条件，而内部节点是回退、序列或并行。

Node type	Succeeds	Fails	Running
Fallback	If one child succeeds	If all children fail	If one child returns running
Sequence	If all children succeed	If one child fails	If one child returns running
Parallel	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	Upon completion	When impossible to complete	During completion
Condition	If true	If false	Never

图 1: 结点类型

当行为树被执行时，根节点以给定的频率和相应的时间步长被激活。然后，该激活信号将按照不同节点类型的规则在树中向下移动，直到到达叶节点。在那里，进行一些计算，通常同时考虑内部状态和传感器数据。如果叶节点是一个行为结点，它可能会向机器人执行器发出一些命令，并将 Success、Failure 或 Running 返回给其父节点。然后，父节点要么向其父节点返回相同的消息，要么选择激活另一个子节点，后者依次返回 Success、Failure 或 Running，以此类推。

接下来将介绍本文中列举的所有行为树结点的功能描述。

3.1 回退结点

当一组行动代表达成类似目标的替代方式时，使用回退结点。因此，该类型结点将从左到右尝试其每个子项，并在找到返回成功的子项后立即返回 Success。只有在所有孩子都失败时才返回 Failure，它就会返回 Running。伪代码如图 2所示。

Algorithm 1: Pseudocode of a Fallback node with N children.

```
for i ← 1 to N do
    childStatus ← Tick(child(i))
    if childStatus = running then
        return running
    else if childStatus = success then
        return success
return failure
```

图 2: 回退节点伪代码

3.2 序列结点

当某些操作需要按顺序执行时，以及当一个操作的成功需要执行下一个操作时，使用序列结点。因此，该结点查找并执行第一个未返回成功的子项。当序列的一个子项返回 Failure 或 Running 时，序列将立即返回状态 Failure 或 Running。伪代码如图 3所示。

Algorithm 2: Pseudocode of a Sequence node with N children.

```
for i ← 1 to N do
    childStatus ← Tick(child(i))
    if childStatus = running then
        return running
    else if childStatus = failure then
        return failure
return success
```

图 3: 序列节点伪代码

3.3 并行结点

并行节点同时激活其所有子节点。如果 N 个子节点中的 M 个子节点返回成功，那么并行节点也会返回 Success。如果大于 N− M 返回失败，从而导致无法成功，它返回 Failure。如果以上条件均不满足，则返回 Running 状态。伪代码如图 4所示。

Algorithm 3: Pseudocode of a parallel node with N children and success threshold M.

```
for i ← 1 to N do
    childStatus(i) ← Tick(child(i))
    if Σ_{i:childStatus(i)=success} 1 ≥ M then
        return Success
    else if Σ_{i:childStatus(i)=failure} 1 > N − M then
        return failure
return running
```

图 4: 并行节点伪代码

3.4 行为结点

行为节点执行一个操作，如果操作完成，则返回 Success，如果无法完成则返回 Failure，如果正在完成则返回 Running。

3.5 条件结点

条件节点确定是否满足给定条件，因此，成功或者失败通常被解释为真或假。

4 复现细节

本人所在课题组负责构建高效迅速的系统平台统筹多机器人智能化感知场景，多机器人控制等，实现灾难场景下的智能化感知、决策、控制。简单来说就是设计一个系统，让机器人可以在火灾场景下进行一些例如侦察、救援的任务。主要工作就是把行为树模型应用在了系统各个模块的任务生成，以及机器人任务的执行。实现了文章中讲到的五个结点，并且为了更加切合系统的实际需求，另外新增了两种类型的控制结点，分别为优先级结点和重复节点。

4.1 与已有开源代码对比

参考了 ROS 平台上提供的行为树在 ROS 系统下的应用例子。将行为树模型移植到我们救援系统上的任务生成模块以及机器人任务执行模块中。具体是代码实现了论文中介绍到的五种类型的结点，在设计具体任务时，调用行为树模块进行任务的插入或删除，在机器人执行任务时，动态地遍历行为树，从而找到下一个需要执行的任务。除此之外，新增了下面展示的两种控制结点。

4.1.1 优先级结点

为了实现动态地插入任务，来面对一些突发情况，例如导航时遇到障碍，或者侦察时遇到所困者。因此，设计出了优先级结点。它可以根据任务子树的优先级插入到对应行为树的位置，所有子任务返回成功才返回 Success。若其中一个子任务执行失败则返回 Failure。具体例子如下图 5所示。机器人在执行救援任务下的导航任务时，由于遇到了障碍，导致执行任务的阻塞，系统会自动生成一个优先级会比导航任务高的除障任务，由于救援任务是由优先级结点控制，所以当插入除障任务时，会根据任务的优先级进行插入。由于除障任务的优先级高于导航任务，所以会将除障任务插入到导航任务之

前。由于优先级结点在激活子任务时是按照从左往右的顺序激活，所以会优先执行除障任务。

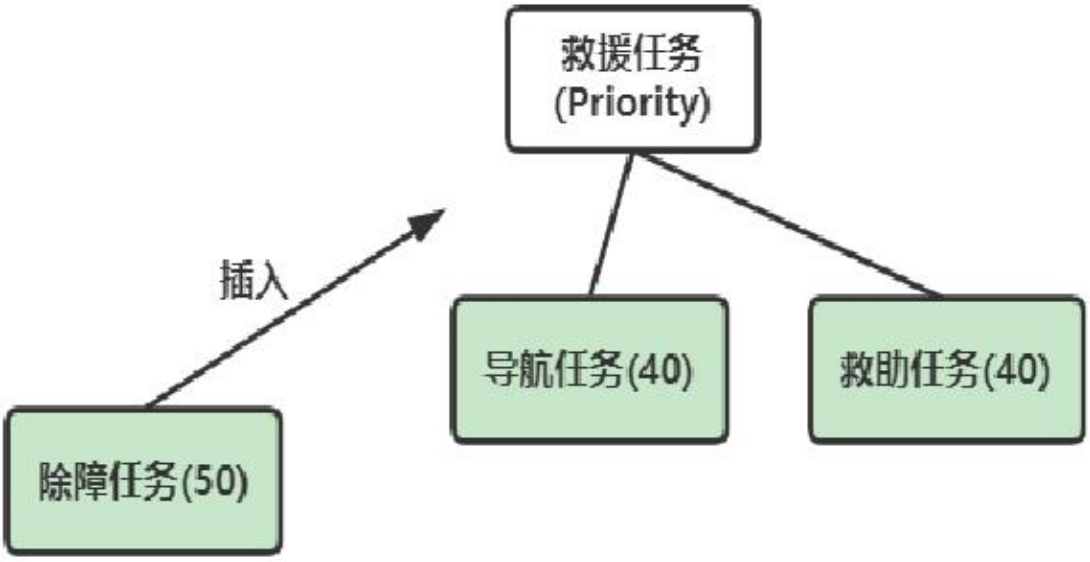

图 5: 优先级结点例子

4.1.2 重复结点

在此之前，某些子任务失败将导致总的任务失败，如果重新执行总的任务会导致某些已经成功执行的子任务会重复地执行，造成资源的浪费。因此，设计出了重复节点。某些子任务执行失败后可以立即重新执行，重新执行的次数由后台控制人员决定。如果在重复执行的次数超过指定次数，且执行失败，则返回 Failure。如果在指定次数内，执行成功，则返回 Success。具体例子如下图 6所示。在执行资源保障任务时，拾取任务由于某些原因导致失败，由于资源保障任务是由重复结点控制，所以拾取任务会立即重新执行，而不会直接返回 Failure, 从而令整个资源保障任务失败。

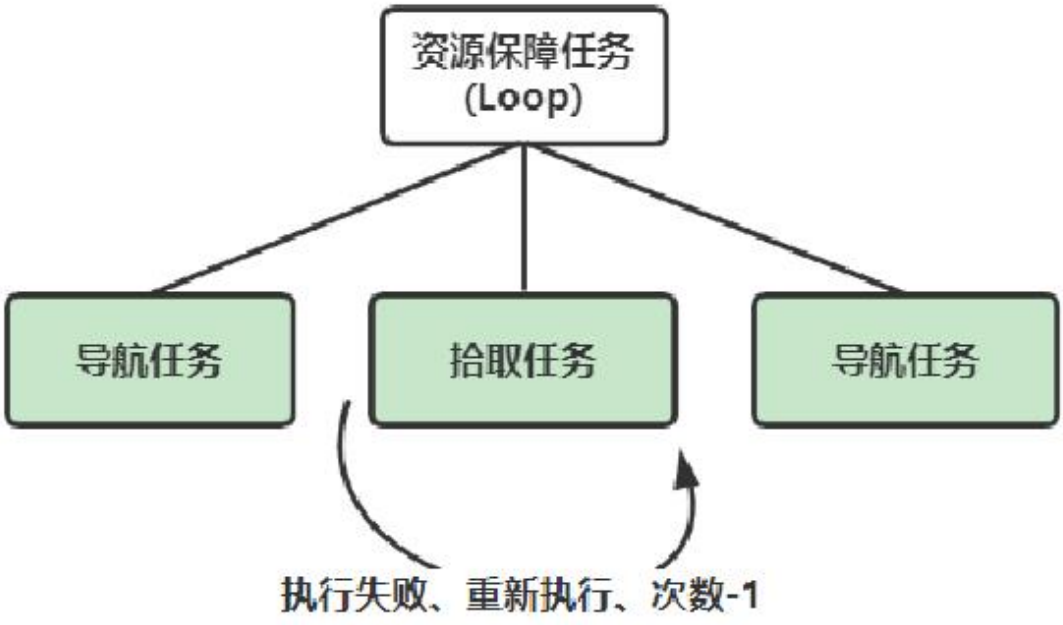

图 6: 重复结点例子

4.1.3 五种控制结点的具体应用

如图 7所示，这个是仿真过程中遇到障碍时机器人的行为树。由于机器人本身维护一个优先级结点的行为树，所以生成的救援任务会插入在救援导航任务的前面。而除障方案执行任务只需一个方案执行成功则成功，所以它的结点是回退结点。而自带工具破拆任务，用到了序列结点，所以所有子任务依次执行成功才，返回成功。另外，寻找工具破拆任务用到了重复结点，这样子任务执行失败可以重新执行。由于导航点的选取涉及到好多因素，包括能否到达，是否在同一区域、物品机器人的大小等。所以现在是目标点附近直接生成四个导航点，其中一个成功到达，则导航成功，所以导航任务用到了回退结点。

图 7: 遇障时机器人的行为树

为了模拟并行结点的应用，设计出了一个开门任务。开门的过程是由机器人一边走一边扭门把手的行为组成，具体例子如下图 8所示。

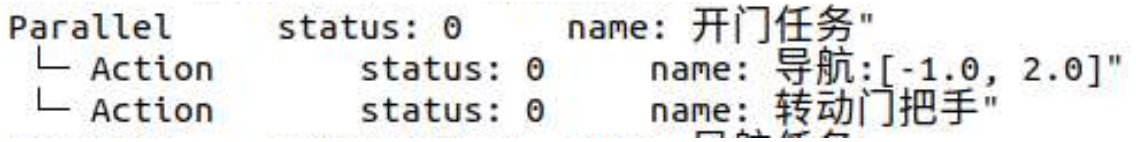

图 8: 实验平台及系统

4.2 实验环境搭建

实验用到了如下 ubuntu18(操作系统)、ros(机器人操作系统 melodic 版本) 、机器人 (turtlebot3)、Gazebo(仿真平台) 几个平台及系统。

图 9: 实验平台及系统

4.3 界面分析与使用说明

仿真界面如下图 10所示，左边是 gazebo 仿真平台显示的内容，实时显示机器人在火灾救援场景下进行的一系列的任务。左边三个控制台，从上到下分别显示机器人的行为树变化的情况、机器人侦察现场的情况、人机交互的输入输出。

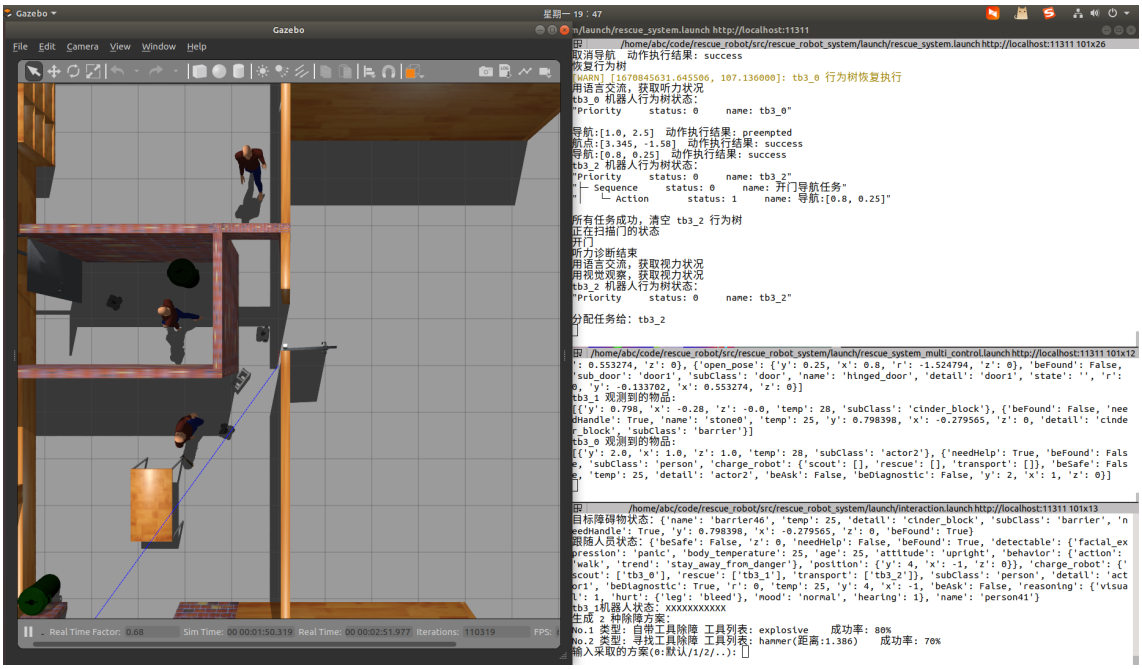

图 10: 实验平台及系统

4.4 创新点

1、将行为树模式应用到课题组所设计的救援系统中，用于机器人任务的生成及执行过程，来辅助机器人面对突发事件的处理，提高执行效率。

2、除了文章提到的五种结点外，另外设计了两种控制结点，更加贴合系统实际的需求。

3、典型的行为树是整棵树事先创建好的，而我们所设计的行为树是可以动态进行插入、修改等操作，这样可以更加高效地应对不确定的情况。

5 实验结果分析

附带的文件中包含了详细的演示视频，视频展示的是，在系统运行时，会根据一些突发情况去生成任务，分配给机器人去执行。例如，一开始执行侦察模块，会分配一些巡逻任务给侦察型机器人。侦察型机器人在执行巡逻任务的时候，侦察现场的情况，并且遇到所困者或者障碍会将具体信息上传到参数服务器。救援模块就会根据这些信息去分配一些救援任务或者运输任务给对应的救援型机器人。除障模块就会生成除障任务，分配给对应的除障机器人。最后，所有巡逻任务执行成功，且场景内所

有受困者成功撤离，则系统结束。在使用行为树之前，机器人执行任务用到的是优先队列，行为树在可读性、可重用性、可用性和编码难度等方面都由于优先队列。使用行为树，在打印时可以直观的看出机器人任务执行的状态，一些任务模块可以重复地调用，任务的设计也比之前简单。

	优先队列（之前）	行为树
可读性	弱	强
可重用性	弱	强
模块化	弱	强
编码难度	难	较易

图 11: 对比

6 总结与展望

在行为树结点应用情况方面，由于并行结点要求子任务同时执行，并且子任务间是相互独立的，之间不能存在一些资源信息的交换共享，所以在任务的设计时，较少地用到了该类型的结点。并且在使用并行结点时，可能会出现数据不一致的情况，从而导致系统死机的情况。系统中用行为树所构造的任务模块，是事前写好的。希望在日后，能够通过强化学习这类的技术，在系统运行过程中，动态地生成满足当前状况的行为树，从而降低人的工作量。

参考文献

[1] COLLEDANCHISE M, ÖGREN P. How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees[J]. IEEE Transactions on robotics, 2016, 33(2): 372-389.