Version: 4.0.1

Remapping ports of a SubTrees

In the CrossDoor example, we saw that a `SubTree` looks like a single leaf Node from the point of view of its parent tree.

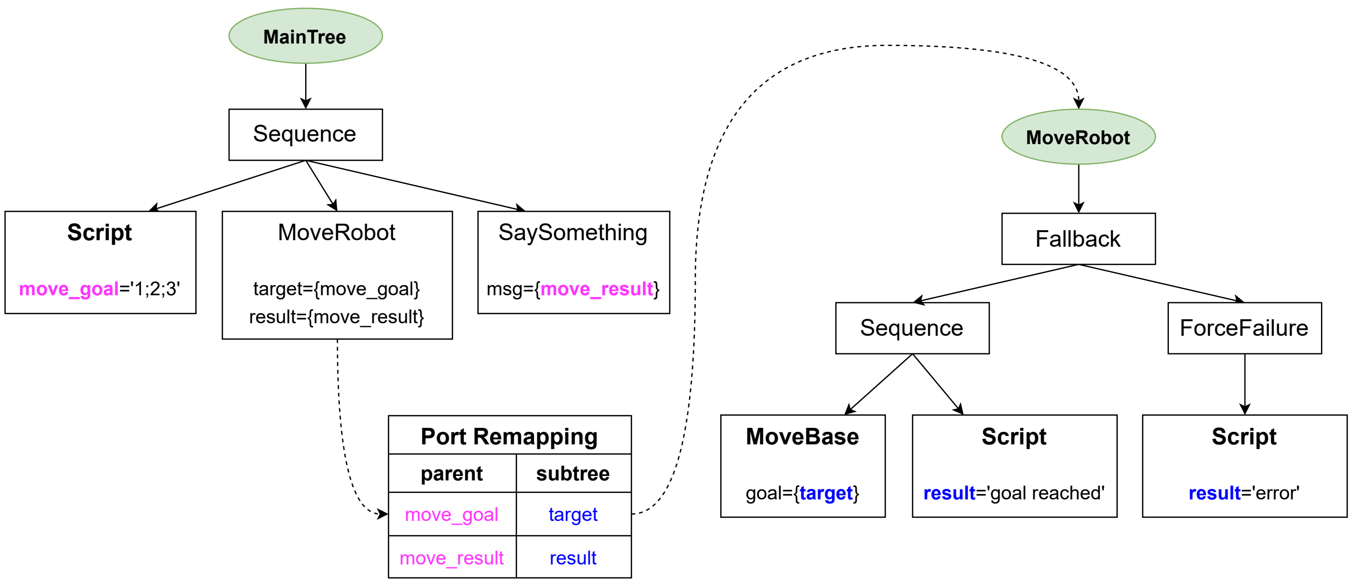
To avoid name clashing in very large trees, any tree and subtree use a different instance of the Blackboard.

For this reason, we need to explicitly connect the ports of a tree to those of its subtrees.

You **won't** need to modify your C++ implementation since this remapping is done entirely in the XML definition.

Example

Let's consider this Behavior Tree.



```
<root BTCPP_format="4">
```

```
    <BehaviorTree ID="MainTree">
        <Sequence>
            <Script script=" move_goal='1;2;3' " />
            <SubTree ID="MoveRobot" target="{move_goal}"
                                    result="{move_result}" />
            <SaySomething message="{move_result}"/>
        </Sequence>
    </BehaviorTree>

    <BehaviorTree ID="MoveRobot">
        <Fallback>
            <Sequence>
                <MoveBase  goal="{target}"/>
                <Script script=" result:='goal reached' " />
            </Sequence>
            <ForceFailure>
                <Script script=" result:='error' " />
            </ForceFailure>
        </Fallback>
    </BehaviorTree>

</root>
```

You may notice that:

- We have a `MainTree` that includes a subtree called `MoveRobot`.
- We want to "connect" (i.e. "remap") ports inside the `MoveRobot` subtree with other ports in the `MainTree`.
- This is done with the syntax used in the example above.

The CPP code

Not much to be done here. We use the `debugMessage` method to inspect the value of the blackboard.

```
int main()
{
  BT::BehaviorTreeFactory factory;

  factory.registerNodeType<SaySomething>("SaySomething");
  factory.registerNodeType<MoveBaseAction>("MoveBase");
  factory.registerBehaviorTreeFromText(xml_text);
  auto tree = factory.createTree("MainTree");

  // Keep ticking until the end
  tree.tickWhileRunning();

  // let's visualize some information about the current state of the
blackboards.
  std::cout << "\n------ First BB ------" << std::endl;
  tree.subtrees[0]->blackboard->debugMessage();
  std::cout << "\n------ Second BB------" << std::endl;
  tree.subtrees[1]->blackboard->debugMessage();

  return 0;
}

/* Expected output:

------ First BB ------
move_result (std::string)
move_goal (Pose2D)

------ Second BB------
[result] remapped to port of parent tree [move_result]
[target] remapped to port of parent tree [move_goal]

*/
```

Edit this page