Behavior Tree Based Simulated Command and Control

Noah Syrkis

June 26, 2025

1 | Granular decision making

2 | Domain Specific Language

1 | Granular decision making

- Control problems are inherently hierarchical
- An order from a general commanding the task of a betalion is high-level
- In action of each individual solder is not specified there
- The simulation strategic encounters has been saturated with deep learning algorithms
- [1]

2 | Domain Specific Language

- The domain specific language (DSL) shown in Table 1 specifies unit behavior
- Figure 1 shows the grammat used to specify behavior

root	→	*tree* (▷ *tree*)⋆	program
tree	→	*leaf* | *node*	node or leaf
leaf	→	A (*move* | *attack*)	action
	|	C (*reachable*)	condition
node	→	S (*root*)	sequence
	|	F (*root*)	fallback
move	→	*move direction*	move action
direction	→	to | from	direction

Table 1: Behavior tree grammar

2 | Domain Specific Language

```
F (
  A move target ▷
  S (
    C is_alive ▷ A move target
  ) ▷
  A move target,
)
```

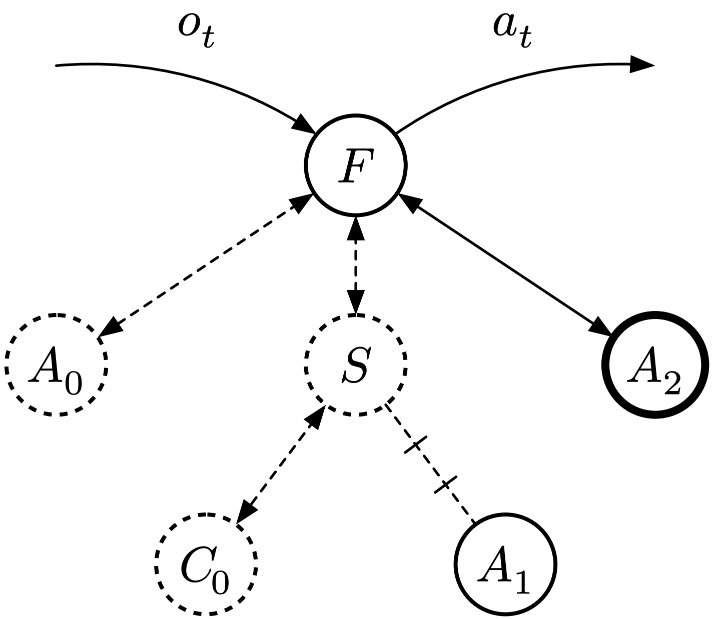

Figure 1: Behavior tree example (see Appendix A for implementation details)

References

[1] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. 2018. doi: 10.1201/9780429489105.

A | Trees vs. Arrays

- Trees versus arrays
- Leaf nodes are actions and conditions
- Rest are sequence or fallback combinators

A | Trees vs. Arrays

index	**parent**	**follows**	**skips**
A_0	F	–	3
C_0	S	S	1
A_1	S	F	1
A_2	F	S	0

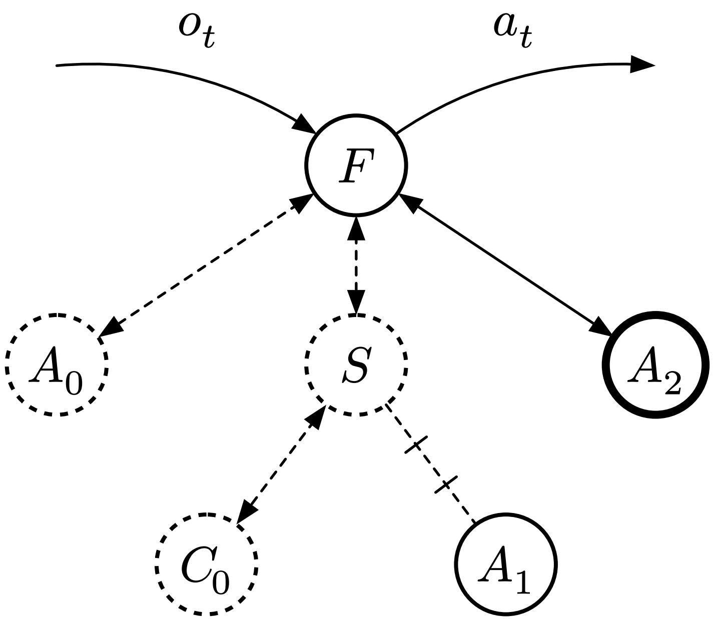

Figure 2: Behavior tree (left) and corresponding behavior array (right)