Attentional Trees And Robotics: Towards an Executive Framework Meeting High-level Decision Making and Control Layer Features.

[st] Pilar de la Cruz
Intelligent And Interactive Systems Department
University of Innsbruck
Innsbruck, Austria
pilar.de-la-cruz@uibk.ac.at

[nd] Matteo Saveriano
Intelligent And Interactive Systems Department
University of Innsbruck
Innsbruck, Austria
matteo.saveriano@uibk.ac.at

[nd] Justus Piater
Intelligent And Interactive Systems Department
University of Innsbruck
Innsbruck, Austria
justus.piater@uibk.ac.at

***Abstract*—(Brainstorm) Motivation: HRI. Attentional robot - flexible, if possible event-based. Introduce behavior tree trend. Potential in robotics, yet too high level. Merge. JSON. What is this system providing as new? Why is it convincing?**

***Index Terms*—behavior trees, robotics, event-based, json**

I. Introduction

A. Behavior Trees (Brainstorm)

- Ticking - the ability to tick allows for work between executions without multi-threading
- Priority Handling - switching mechansims that allow higher priority interruptions is very natural
- Simplicity - very few core components, making it easy for designers to work with it
- Dynamic - change the graph on the fly, between ticks or from parent behaviours themselves
- Self-explainable

B. Behavior Trees Limitations (Brainstorm)

C. How Limitations can be overcome: JSON (Brainstorm)

JSON is a standard format for data exchange inspired by JavaScript. Generally, JSON is in string or text format. (EXTEND overview) JSON schemas provide (1) simplicity due to its compact description of information, (2) flexibility due to its abstract base representation of tasks and (3) modularity due to the definition of subtrees which can be added and/or removed on the fly in whatever position of Behavior Tree. Finally, JSON support with databsaes such as mySQL and MongoDB motivates the fact of bringing up a general string schema which couples not only task executions but also an abstract knowledge representation of actions.

D. What is this document about (Brainstorm)

E. Describe Sections Structure and Content

II. Related Works

Add Matteo's system. Add SoA Behavior Trees.

- Matteo's system: Refer to Method's section.

III. System Overview

Attentional Trees framework aims at integrating the following features:

1) Task planner based on Behavior Trees
2) Priority handling of tasks based on attentional mechanisms and emphasis [2]
3) JSON schema conversion into Behavior Trees

A. Task Planning

B. Priority Handling

1) Attentional Mechanism:

2) Emphasis:

C. JSON Schema

A typical JSON schema consists of a list of string objects. The first object element defines the root of the Behavior Tree. The remaining object elements are defined as subtrees and will be added recursively as children from the root. Each object contains properties `name`, `type`, `father`, `children` and `parameters`, which provide with information about the Behavior Tree node types as well as additional attributes for pre-and post-conditioning handling, priority execution and any additional information.

```
{
    "name":"selector01",
    "type":"selector",
```

```
    "father": "sequence01",
    "children": ["isReached","reach"],
    "parameters": [["emphasis", 1]]
}
```

D. Event-Based Behavior Tree

IV. Implementation

The framework platform of Attentional Trees is based on the open-source library py_trees [4]. py_trees is compatible with ROS and provides the easiness and quickness of an implementation based in python language. Moreover, decoding operations becomes specially intuitive thanks to the in-built mapping between container and value (JSON objects to Python objects and viceversa) based on Python standard libraries. *Note: justify the contribution of our work to the library

A. py tree Overview (Brainstorm)

- idioms for PA-BT creation
- blackboard and emphasis for attentional mechanism
- trees and behavior classes
- demo overviews

B. Adapting Procedure (Brainstorm)

- reasoning of applied demos as basis
- use of idioms for PA-BT creation approach
- use of blackboard for emphasis parameter and priority handling
- reasoning of behaviors' usage for simulations
- read of schema variables and mapping for pre-condition, post-condition and blackboard variables

V. Simulations

In order to test the requirements described in section above, the following demo programs have been implemented[1]

- `json-tree.py`. This program focuses on (1) loading JSON schema and (2) mapping into a Behavior Tree based on PA-BT approach [3]. *Note: currently (8.12.2019) PA-BT is extended via `automated-panning.py` demo. `json-tree.py` provides with a Behavior Tree extension following a minimal implementation and no use of idioms.
- `attentional-tree.py`. This program focuses on high-priority handling based on access to emphasis value [1].

Discussion

References

[1] Add Matteo's paper and page of emphasis description
[2] https://www.guru99.com/python-json.html
[3] Behavior Tree book
[4] https://py-trees.readthedocs.io/en/devel/

[1]These are available on: