C2SIM

Noah Syrkis
IT University of Copenhagen
Apr. 17, 2024

1 | Overview

The project[1] uses JAX[2] throughout, with JaxMARL's[3] [1] StarCraft II-like SMAX as the environment. The agents are modelled using behaviour trees (BT). BTs are defined using a domain specific language (DSL) developped for the purpose. The ollama[4] library is used for the language modelling to map game states to human language and BTs, and vice versa.

[1] https://github.com/syrkis/c2sim/

[2] https://github.com/google/jax/

[3] https://blog.foersterlab.com/jaxmarl/

[4] https://ollama.com/

1 | Overview (cont.)

- ☒ SMAX visual playback (`src/{plot,smax}.py`).
- ☒ BT function constructor (`src/{bt,atomics}.py`).
- ☒ BT based trajectory (`src/smax.py`). (yet to JIT compile)
  - ▶ Must traverse all leafs always (for array programming)[5].
- ☒ Domain specific language (`grammar.lark`).
- ☐ Implement the BTBank (`src/bank.py`).
- ☐ Language out (`src/llm.py`).
- ☐ Language in (`src/llm.py`).

[5] Has no effect on performance, as we are always as slow as slowest action

2 | SMAX

- Extensive work on visual playback of trajectory fig. 1.
  - ☒ Costum SMAX vizualization.
  - ☒ Show unit type, team, health, attacks, and reward.
  - ☒ Successfully runnning 10K+ parallel environments.

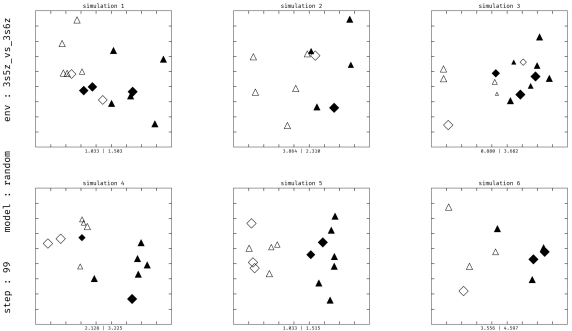

Figure 1: SMAX in parallel

3 | Behaviour trees

- BT is for now is located in a .yaml file.
- Beginning move to sqlite3 database.
- JAX based tick functions for node and leafs.
- Full traversal happens every tick, using logical operations.
- No JIT compilation yet.

3 | DSL grammar

```
1 tree      : sequence | fallback | decorator | atomic
2 atomic    : action | condition
3 nodes     : tree ( :: tree )*
4 sequence  : S ( nodes )
5 fallback  : F ( nodes )
6 decorator : D ( nodes )
7 action    : A ( STRING+ )
8 condition : C ( STRING+ )
```

3 | DSL example

```
S (
  S (
    C ( see enemy_0 ) :: A ( attack enemy_0 )
  ) ::
  F (
    C ( see_enemy ) :: A ( find_enemy )
  ) ::
  A ( attack_enemy )
)
```

3 | Atomics

- Atomics are the leaves (actions/conditions) of the tree.
- They are JAX functions.
- Keep them simple and fast (complex behavior should come from the tree).
  - E.g. `move`, `attack`, `is_enemy`, `is_dead`, `n_in_range`, etc.
  - Maybe map out desired atomic functions.

3 | BTBank

- BTBank is a library for creating and running BTs.
- It is written in Python.
- sqlite3 is used to store the trees.

4 | Language model

- The language model is a transformer model.
- I/O architecture.
- The output is a sequence of tokens.

References

[1] Alexander Rutherford et al. *JaxMARL: Multi-Agent RL Environments in JAX*. Dec. 2023. DOI: 10.48550/arXiv.2311.10090. arXiv: 2311.10090 [cs].