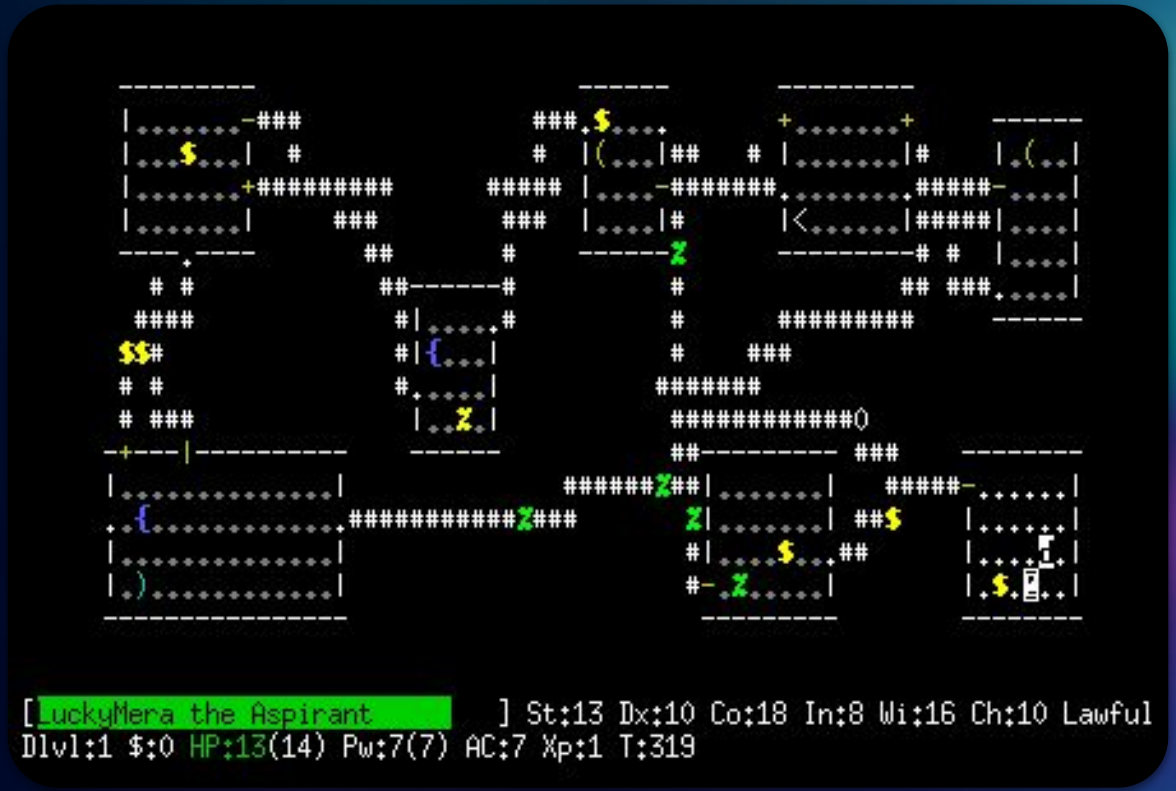
AI vs NetHack

Luigi Quarantiello
luigi.quarantiello@phd.unipi.it

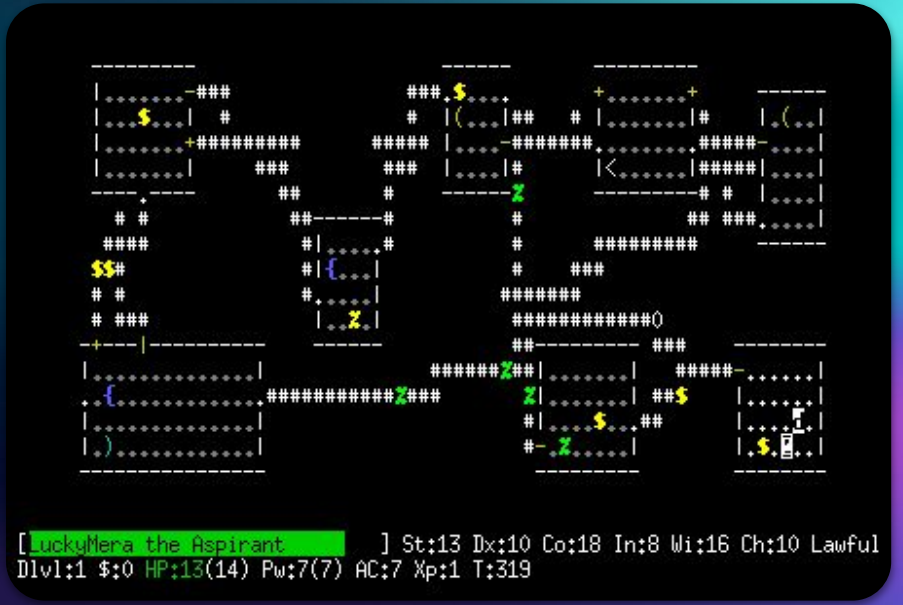
Simone Marzeddu
s.marzeddu@studenti.unipi.it

What is NetHack?



What is NetHack?

- NetHack is a roguelike video game
- Hundreds of objects and monsters
- Several roles, races and alignments



The NetHack Wiki

Why NetHack?

- AI Research needs testbeds!
- Often based on simulated environments, like video games!

Why NetHack?

- Complex, rich, procedurally generated, stochastic open world
- Fast simulator
- Perfect to test generalization capabilities

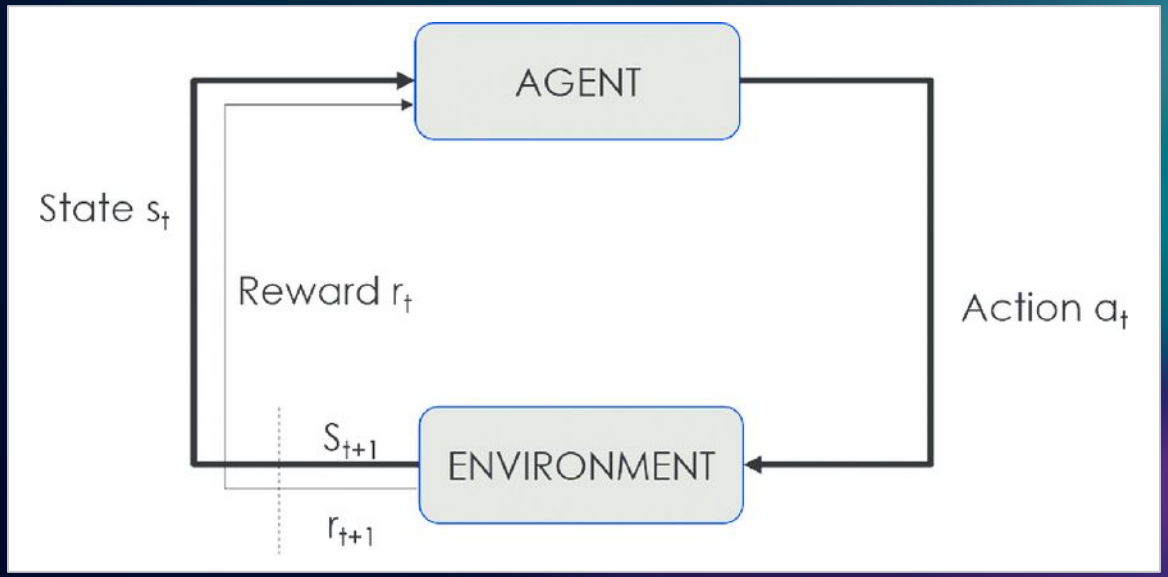
- Still an open challenge!

Learning in NetHack

- RL environment
- Provides all the observations and actions from the game

The nethack learning environment. Küttler et al, 2020.
Openai gym, Brockman et al, 2016.
Code example

RL!?!?!?



Reinforcement Learning cycle

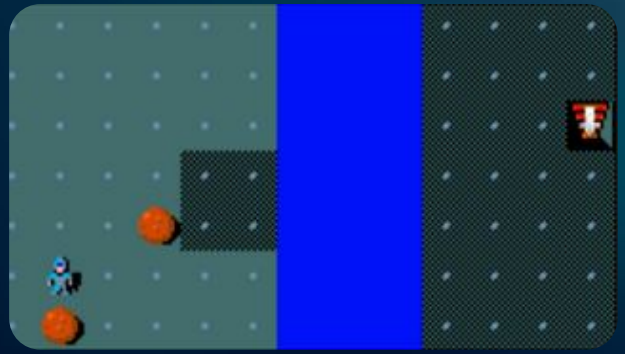
Learning in NetHack

- Built on top of NLE
- Offers a suite of simpler tasks

Minihack the planet, Samvelyan et al, 2021

Learning in MiniHack

MazeWalk



River

Room
Ultimate

Minihack the planet, Samvelyan et al, 2021

Learning in MiniHack

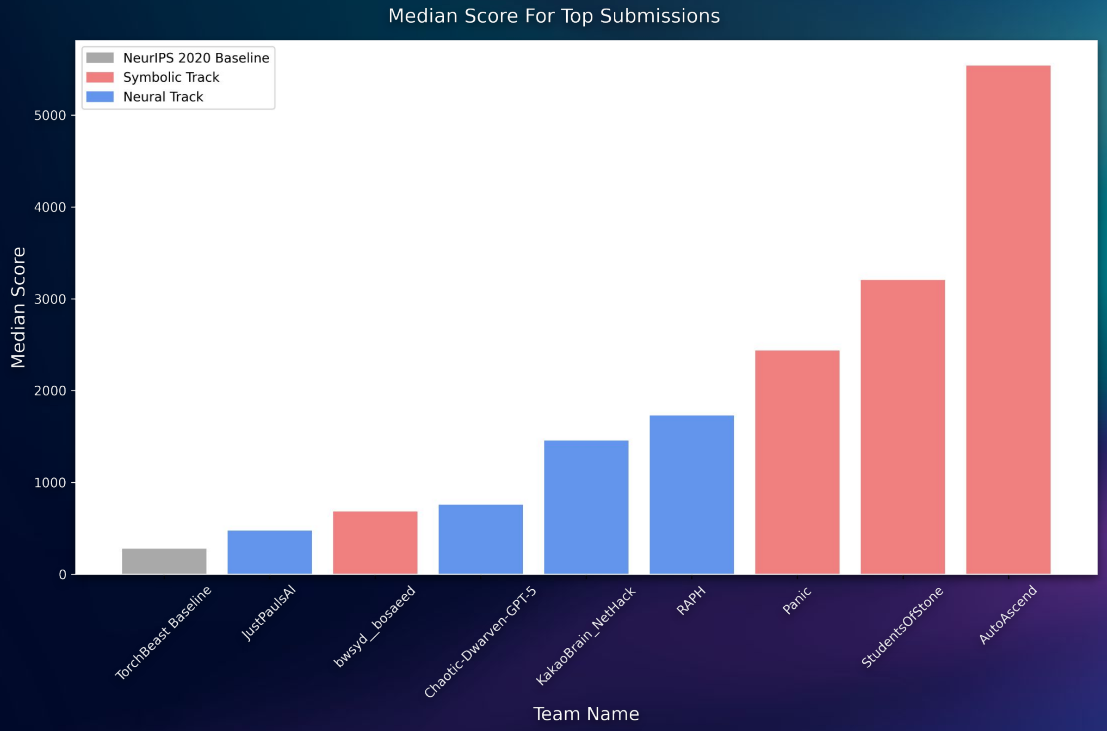
MiniHack Level Editor

NeurIPS 2021 NetHack Challenge

- Build an agent that can beat the game and win 3000$!
- Prizes for both neural and symbolic agents
- Also open to students and researchers

NeurIPS 2021 NetHack Challenge

NeurIPS 2021 NetHack Challenge



NeurIPS 2021 NetHack Challenge

Applications in NetHack

LuckyMera

A Modular AI Framework for Building Hybrid NetHack Agents

LuckyMera

A Modular AI Framework for Building Hybrid NetHack Agents

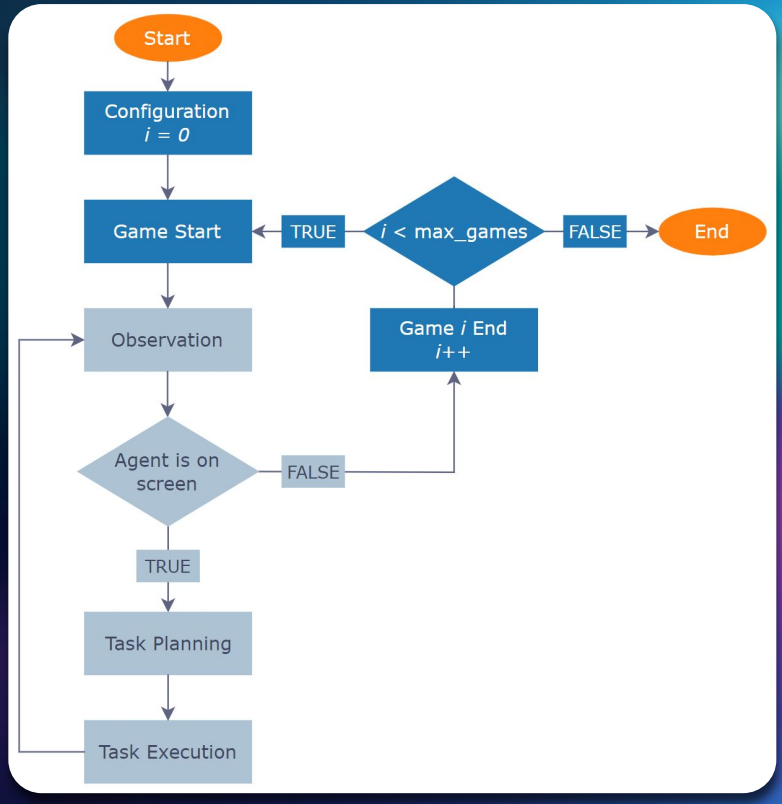
- Complete framework to train, integrate and test different AI approaches
- Well-suited for symbolic, neural and hybrid models
- Easy to implement new solutions!

LuckyMera

LuckyMera

How does it works?

- Iterative execution of the highest-priority plannable skill
- Two main phases:
  - skill-planning
  - skill-execution



LuckyMera

3 basic features - Skill Definition

Implement new modules by extending:

Skill

- Base class
- Only general-purpose methods

ReachSkill

- Navigation tasks

HiddenSkill

- Methods to find hidden passages and areas

LuckyMera

3 basic features - Skill Integration

- LuckyMera comes with the implementation of several skills, defining a cautious agent, but...
- You can import any external module
  - Define new actions
  - Override the existing behavior, with a different strategy/AI model

LuckyMera

3 basic features - Training

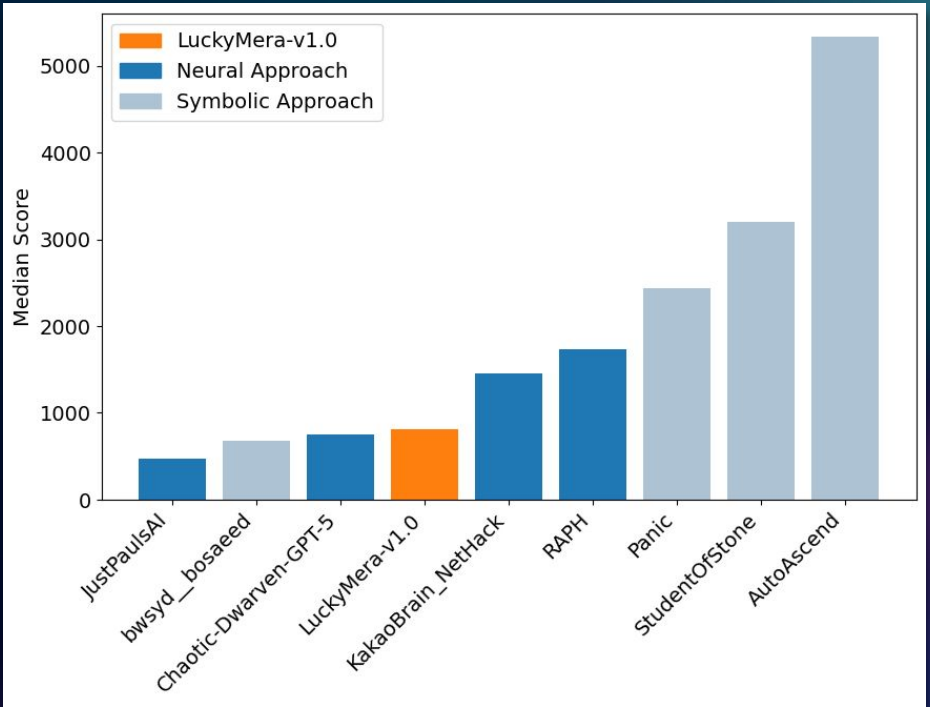
- Integrated training processes
- Easy to test neural models
- Integration with other skills, to define hybrid approaches

Algorithm 1 *Behavioural Cloning*

Ensure: a *policy* π_θ trained on the problem
  while $L(a^*, \pi(s))$ is not small enough **do**
    Collect trajectories $\tau_1, \ldots, \tau_n$ from the expert.
    Get all the (s_i^*, a_i^*) from each τ_i, as *i.i.d.* pairs.
    Learn *policy* π^* by minimizing $L(a^*, \pi(s))$.
  end while

LuckyMera

Empirical Evaluation



Knowledge Base for NetHack

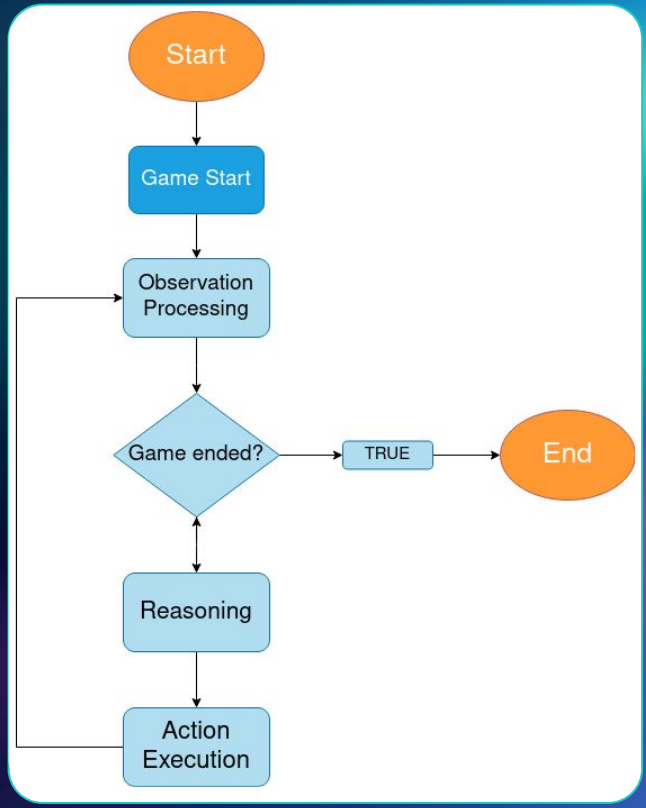
PING

Introduction to Prolog

- One of the most popular logic programming languages
- It uses first-order logic clauses to define a knowledge base
- Problems are solved by querying the *kb*
- Particularly suited for Artificial Intelligence applications

Knowledge base for NetHack

- **Python** scripts for the low-level interaction with the environment
- **Prolog** program for the reasoning process



The environment

- MiniHack custom level
- 10x10 grid, with a monster, a weapon, a trap and an apple
- Eat the apple to win

Prolog knowledge base

- It decides the action to perform
- `assert` and `retract` to receive the current status of the world
- Rules in the form of:

```
action(X) :- [body of the rule]
```

- Python-Prolog interaction with PySwip

PySwip

Prolog knowledge base

```
action(attack(Direction)) :- position(agent, _, AgentR, AgentC), position(enemy, Type, EnemyR, EnemyC),
                             wields_weapon(agent, Weapon), is_beatable(Type, Weapon),
                             is_close(AgentR, AgentC, EnemyR, EnemyC), healthy,
                             next_step(AgentR, AgentC, EnemyR, EnemyC, Direction).
```

```
action(get_to_weapon(Direction)) :- position(agent, _, AgentR, AgentC), position(enemy, Type, EnemyR, EnemyC),
                                    is_close(AgentR, AgentC, EnemyR, EnemyC),
                                    wields_weapon(agent, Weapon), \+ is_beatable(Type, Weapon),
                                    position(weapon, tsurugi, WeaponR, WeaponC),
                                    next_step(AgentR, AgentC, WeaponR, WeaponC, D), safe_direction(AgentR, AgentC, D, Direction).
```

```
safe_direction(R, C, D, Direction) :- resulting_position(R, C, NewR, NewC, D),
                                      ( safe_position(NewR, NewC) -> Direction = D;
                                      % else, get a new close direction
                                      % and check its safety
                                      close_direction(D, ND), safe_direction(R, C, ND, Direction)
                                      ).
```

Neuro-Symbolic Approaches

Neural Approach with IMPALA

- *KeyRoom-S5* on MiniHack
- The model was trained for 20 million steps
- Mean return of 0.99

Torchbeast: A pytorch platform for distributed rl, Küttler et al, 2019

Neural Approach with IMPALA

- *Room-Ultimate-15x15* on MiniHack
- The model was trained for 20 million steps
- Mean return of 0.75

Torchbeast: A pytorch platform for distributed rl, Küttler et al, 2019

Integrating the rules

- Combine the neural model with a rule-based system
- Better results and efficiency
- Some examples:
  - `rule_attack_monster`
  - `rule_do_not_hit_walls`
  - `rule_do_not_repeat_action`
  - `rule_explore`

Integrating the rules

Room-Ultimate-15x15

- Without the rules, mean return of 0.75
- With two rules, mean return of 0.9!

Keyroom-S5

- Without rules and 5e6 steps, mean return of 0.09
- With two rules, mean return of 0.72!

Thanks for your attention!!

luigi.quarantiello@phd.unipi.it

You can find me @ Room 390!