

gym-saturation: an OpenAI Gym environment for saturation provers

Boris Shminke¹

¹ Laboratoire J.A. Dieudonné, CNRS and Université Côte d'Azur, France

DOI: [10.21105/joss.03849](https://doi.org/10.21105/joss.03849)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Viviane Pons](#) ↗

Reviewers:

- [@lutzhamel](#)
- [@lan-bunner](#)

Submitted: 01 October 2021

Published: 20 November 2021

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

gym-saturation is an OpenAI Gym ([Brockman et al., 2016](#)) environment for reinforcement learning (RL) agents capable of proving theorems. Currently, only theorems written in a formal language of the Thousands of Problems for Theorem Provers (TPTP) library ([Sutcliffe, 2017](#)) in clausal normal form (CNF) are supported. gym-saturation implements the 'given clause' algorithm (similar to the one used in Vampire ([Kovács & Voronkov, 2013](#)) and E Prover ([Schulz et al., 2019](#))). Being written in Python, gym-saturation was inspired by PyRes ([Schulz & Pease, 2020](#)). In contrast to the monolithic architecture of a typical Automated Theorem Prover (ATP), gym-saturation gives different agents opportunities to select clauses themselves and train from their experience. Combined with a particular agent, gym-saturation can work as an ATP. Even with a non trained agent based on the clause size heuristic, gym-saturation can find refutations for 2245 (of 8257) CNF problems from TPTP v7.5.0.

Statement of need

Current applications of RL to saturation-based ATPs like Enigma ([Jakubuv et al., 2020](#)) or Deepire ([Suda, 2021](#)) are similar in that the environment and the agent are not separate pieces of software but parts of larger systems that are hard to disentangle. The same is true for non saturation-based RL-friendly provers too (e.g. lazyCoP, [Rawson & Regeer \(2021\)](#)). This monolithic approach hinders free experimentation with novel machine learning (ML) models and RL algorithms and creates unnecessary complications for ML and RL experts willing to contribute to the field. In contrast, for interactive theorem provers, some projects like HOList ([Bansal, Loos, Rabe, Szegedy, & Wilcox, 2019](#)) separate the concepts of environment and agent. Such modular architecture proved to help create multiple successful agents by different groups of researchers (see, e.g. [Paliwal et al. \(2020\)](#) or [Bansal, Loos, Rabe, & Szegedy \(2019\)](#)). gym-saturation is an attempt to implement a modular environment-agent architecture of an RL-based ATP. In addition, some RL empowered saturation ATPs are not accompanied with their source code ([Abdelaziz et al., 2021](#)), while gym-saturation is open-source software.

Usage example

Suppose we want to prove an extremely simple theorem with a very basic agent. We can do that in the following way:

```
# first we create and reset a OpenAI Gym environment
from importlib.resources import files
import gym

env = gym.make(
    "gym_saturation:saturation-v0",
```

```

# we will try to find a proof shorter than 10 steps
step_limit=10,
# for a classical syllogism about Socrates
problem_list=[
    files("gym_saturation").joinpath(
        "resources/TPTP-mock/Problems/TST/TST003-1.p"
    )
],
)
env.reset()
# we can render the environment (that will become the beginning of the proof)
print("starting hypotheses:")
print(env.render("human"))
# our 'age' agent will always select clauses for inference
# in the order they appeared in current proof attempt
action = 0
done = False
while not done:
    observation, reward, done, info = env.step(action)
    action += 1
# SaturationEnv has an additional method
# for extracting only clauses which became parts of the proof
# (some steps were unnecessary to find the proof)
print("refutation proof:")
print(env.tstp_proof)
print(f"number of attempted steps: {action}")

```

34 The output of this script includes a refutation proof found:

```

35 starting hypotheses:
36 cnf(p_imp_q, hypothesis, ~man(X0) | mortal(X0)).
37 cnf(p, hypothesis, man(socrates)).
38 cnf(q, hypothesis, ~mortal(socrates)).
39 refutation proof:
40 cnf(_0, hypothesis, mortal(socrates), inference(resolution, [], [p_imp_q, p])).
41 cnf(_2, hypothesis, $false, inference(resolution, [], [q, _0])).
42 number of attempted steps: 6

```

43 Architecture

44 gym-saturation includes several sub-packages:

- 45 ■ parsing (happens during env.reset() in example code snippet)
- 46 ■ logic operations (happen during env.step(action) in the example)
- 47 ■ AI Gym environment implementation
- 48 ■ agent testing (a bit more elaborated version of the while loop from the example)

49 gym-saturation relies on a deduction system of four rules which is known to be refutation
50 complete (Brand, 1975):

$$\frac{C_1 \vee A_1, C_2 \vee \neg A_2}{\sigma(C_1 \vee C_2)}, \sigma = mgu(A_1, A_2) \quad (\text{resolution})$$

$$\frac{C_1 \vee s \approx t, C_2 \vee L[r]}{\sigma(L[t] \vee C_1 \vee C_2)}, \sigma = mgu(s, r) \quad (\text{paramodulation})$$

$$\frac{C \vee A_1 \vee A_2}{\sigma(C \vee L_1)}, \sigma = mgu(A_1, A_2) \quad (\text{factoring})$$

$$\frac{C \vee s \not\approx t}{\sigma(C)}, \sigma = mgu(s, t) \quad (\text{reflexivity resolution})$$

51 where C, C_1, C_2 are clauses, A_1, A_2 are atomic formulae, L is a literal, r, s, t are terms, and
 52 σ is a substitution (most general unifier). $L[t]$ is a result of substituting the term r in $L[r]$
 53 for the term t at only one chosen position.

54 For parsing, we use the LARK parser (Shinan, 2021). We represent the clauses Python classes
 55 forming tree-like structures. gym-saturation also includes a JSON serializer/deserializer for
 56 those trees. For example, a TPTP clause

```
57 cnf(a2,hypothesis,  
58     ( ~ q(a) | f(X) = X )).
```

59 becomes

```
Clause(  
    literals=[  
        Literal(  
            negated=True,  
            atom=Predicate(  
                name="q", arguments=[Function(name="a", arguments=[])]  
            ),  
        ),  
        Literal(  
            negated=False,  
            atom=Predicate(  
                name="=",  
                arguments=[  
                    Function(name="f", arguments=[Variable(name="X")]),  
                    Variable(name="X"),  
                ],  
            ),  
        ),  
    ],  
    label="a2",  
)
```

60 This grammar serves as the glue for gym-saturation sub-packages, which are, in principle,
 61 independent of each other. After switching to another parser or another deduction system,
 62 the agent testing script won't break, and RL developers won't need to modify their agents for
 63 compatibility (for them, the environment will have the same standard OpenAI Gym API).

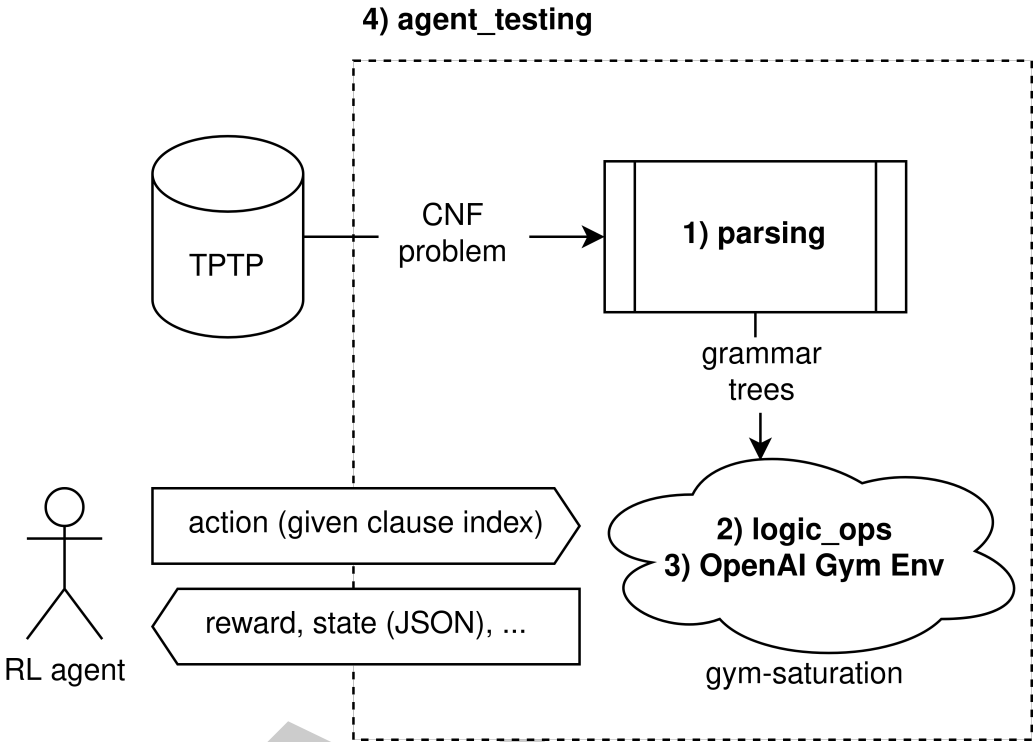


Figure 1: A diagram showing interactions between four main subpackages of gym-saturation: 1) parsing; 2) logic operations (including the given clause algorithm); 3) OpenAI Gym Env implementation; 4) the agent testing script.

64 Agent testing is a simple episode pipeline (see [Figure 1](#)). It is supposed to be run in parallel
65 (e.g. using GNU Parallel, [Tange \(2021\)](#)) for a testing subset of problems. See the following
66 table for the testing results of two popular heuristic-based agents on TPTP v7.5.0 with 20
67 steps limit:

agent	total problems	proof found	step limit reached	error
size	8257	2245	5889	123
age	8257	234	7884	139

68 size is an agent, which always selects the shortest not yet processed clause as an action.
69 age is an agent which chooses clauses in FIFO order. An error means an out-of-memory or
70 one-hour timeout event. Trained RL agents should strive to be more successful than those
71 primitive baselines.

72 Mentions

73 At the moment of writing this paper, gym-saturation was used by its author during their
74 PhD studies for creating experimental RL-based ATPs.

Acknowledgements

This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

References

- Abdelaziz, I., Crouse, M., Makni, B., Austil, V., Cornelio, C., Ikbali, S., Kapanipathi, P., Makondo, N., Srinivas, K., Witbrock, M., & Fokoue, A. (2021). Learning to guide a saturation-based theorem prover. *CoRR*, abs/2106.03906. <https://arxiv.org/abs/2106.03906>
- Bansal, K., Loos, S. M., Rabe, M. N., & Szegedy, C. (2019). Learning to reason in large theories without imitation. *CoRR*, abs/1905.10501. <http://arxiv.org/abs/1905.10501>
- Bansal, K., Loos, S. M., Rabe, M. N., Szegedy, C., & Wilcox, S. (2019). HOList: An environment for machine learning of higher order logic theorem proving. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA* (Vol. 97, pp. 454–463). PMLR. <http://proceedings.mlr.press/v97/bansal19a.html>
- Brand, D. (1975). Proving theorems with the modification method. *SIAM Journal on Computing*, 4(4), 412–430. <https://doi.org/10.1137/0204036>
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. *CoRR*, abs/1606.01540. <http://arxiv.org/abs/1606.01540>
- Jakubuv, J., Chvalovský, K., Olsák, M., Piotrowski, B., Suda, M., & Urban, J. (2020). ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In N. Peltier & V. Sofronie-Stokkermans (Eds.), *Automated reasoning - 10th international joint conference, IJCAR 2020, paris, france, july 1-4, 2020, proceedings, part II* (Vol. 12167, pp. 448–463). Springer. https://doi.org/10.1007/978-3-030-51054-1_29
- Kovács, L., & Voronkov, A. (2013). First-order theorem proving and vampire. In N. Sharygina & H. Veith (Eds.), *Computer aided verification - 25th international conference, CAV 2013, saint petersburg, russia, july 13-19, 2013. proceedings* (Vol. 8044, pp. 1–35). Springer. https://doi.org/10.1007/978-3-642-39799-8_1
- Paliwal, A., Loos, S. M., Rabe, M. N., Bansal, K., & Szegedy, C. (2020). Graph representations for higher-order logic and theorem proving. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2967–2974. <https://aaai.org/ojs/index.php/AAAI/article/view/5689>
- Rawson, M., & Reger, G. (2021). lazyCoP: Lazy paramodulation meets neurally guided search. In A. Das & S. Negri (Eds.), *Automated reasoning with analytic tableaux and related methods - 30th international conference, TABLEUX 2021, birmingham, UK, september 6-9, 2021, proceedings* (Vol. 12842, pp. 187–199). Springer. https://doi.org/10.1007/978-3-030-86059-2_11
- Schulz, S., Cruanes, S., & Vukmirovic, P. (2019). Faster, higher, stronger: E 2.3. In P. Fontaine (Ed.), *Automated deduction - CADE 27 - 27th international conference on automated deduction, natal, brazil, august 27-30, 2019, proceedings* (Vol. 11716, pp. 495–507). Springer. https://doi.org/10.1007/978-3-030-29436-6_29

- 119 Schulz, S., & Pease, A. (2020). Teaching automated theorem proving by example: PyRes
120 1.2 - (system description). In N. Peltier & V. Sofronie-Stokkermans (Eds.), *Automated*
121 *reasoning - 10th international joint conference, IJCAR 2020, paris, france, july 1-4, 2020,*
122 *proceedings, part II* (Vol. 12167, pp. 158–166). Springer. [https://doi.org/10.1007/](https://doi.org/10.1007/978-3-030-51054-1_9)
123 [978-3-030-51054-1_9](https://doi.org/10.1007/978-3-030-51054-1_9)
- 124 Shinan, E. (2021). *Lark-parser* (Version 0.12.0). <https://pypi.org/project/lark-parser/>
- 125 Suda, M. (2021). Improving ENIGMA-style clause selection while learning from history. In
126 A. Platzer & G. Sutcliffe (Eds.), *Automated deduction - CADE 28 - 28th international*
127 *conference on automated deduction, virtual event, july 12-15, 2021, proceedings* (Vol.
128 12699, pp. 543–561). Springer. https://doi.org/10.1007/978-3-030-79876-5_31
- 129 Sutcliffe, G. (2017). The TPTP Problem Library and Associated Infrastructure. From CNF
130 to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4), 483–502.
- 131 Tange, O. (2021). *GNU parallel 20210822 ('kabul')*. Zenodo. [https://doi.org/10.5281/](https://doi.org/10.5281/zenodo.5233953)
132 [zenodo.5233953](https://doi.org/10.5281/zenodo.5233953)

DRAFT