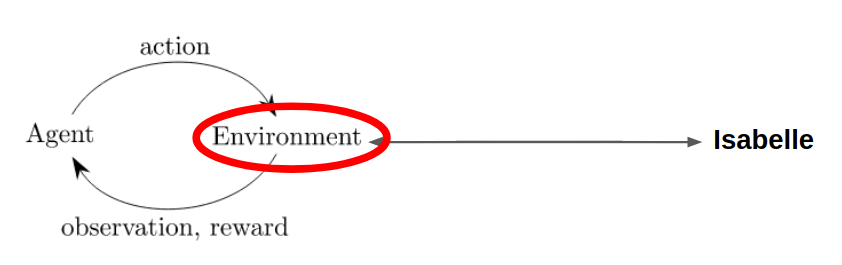
Discussion 1 Record

1. A quick introduction to use of Isabelle:

[Isabelle Theorem Prover tutorial](https://www.youtube.com/watch?v=1nEpUoVopT0)f

1. We need to maintain a tree structure at the agent’s side. Question is whether Isabelle maintains a tree structure at its side and whether we will need to maintain one too. The environment in the red circle is what we need to build.



1. About data
2. TODO: Read documents of Isabelle Server. Docs can be found at this link :

<https://isabelle.in.tum.de/doc/system.pdf>

1. Isabelle Source and instructions:

<https://isabelle.in.tum.de/repos/isabelle>

<https://isabelle.in.tum.de/repos/isabelle/file/12f455cc6573/README_REPOSITORY>

1. A server written by others for HOList. Here is the link:

<https://github.com/jasonrute/holist-communication-example>

Answers to Bando’s questions

1. How would a typical RL environment Gym look like for IsabelleGym? What are the advantages and disadvantages?

A typical RL environment usually has the following characters (take OpenAI/Gym as an [example](https://gym.openai.com/docs/#environments):):

1. Observations
2. Reward
3. A flag showing whether the environment is reset
4. Other info
5. Right now, we need to add an API to communicate between Isabelle and our own environment (sth like a messenger going forward and backward between Isabelle and our environment)
6. Model how an RL agent works with Isabelle. What are the actions. What are the states and the rewards? Where does the proof tree come into the picture? Where does backtracking come into the picture?

Actions : Choose which tactic to take from the set of tactics.

States : A node with a set of subgoals as children. (I think there should be some sort of encoding/decoding method here)

Rewards :

for each level / (or estimate the “distance” from our current state to the empty goal)

The proof tree is the structure/”map” that our agent is going to explore. In the end it either successfully proves the goal by containing a leaf that is an empty goal or fails to prove th goal(with no leaf containing empty goal under the restricted height)

When our agent reaches the restricted height, but still no sense to prove all subgoals correctly, we need to backtrack to find other branches to explore.

1. How does backtracking affect the modeling of an agent (especially an RL agent)? Think about advantages & disadvantages

Advantages: (Also, backtracking might avoid being trapped into local optimal solutions during training?)

* Backtracking is especially useful for RL, since backtracking is a natural action even human agents can take.
* Backtracking is also naturally supported by Isabelle, we can provide this functionality and let the designer of agents decide whether they are allowed or not.

disadvantages

* Probably more computational resources & human efforts - run time, json rewrites.

1. CoqGym and HOList API format. Analyze advantages and disadvantages

All of the APIs has these three main functionalities:

1. initialize the environment/file.
2. Choose theorem to prove.
3. Apply tactic.

Most of the specifications of CoqGym are listed in [CoqGymAPI.md](https://github.com/brando90/isabelle-gym/blob/literature_review/Preview%20Work%20and%20Resources/CoqGymAPI.md).

* we provide dataset
  + - serialized Coq files in json format, we also store a recorded proof tree
* - we provide utilities for communicating with coq
  + - gallina, translator between s-expression and gallina
  + - eval env.py handling interactions between agent and coq(gym), ProofEnv for RL, and FileEnv for supervised learning.
  + - serapi to handle communication between coqgym and coq
* Example usage at [ASTactic/agent.py](https://github.com/princeton-vl/CoqGym/blob/master/ASTactic/agent.py).

Most of the interfaces of HOList are listed in [proof\_assistant.proto](https://github.com/jasonrute/holist-communication-example/blob/master/proof_assistant.proto).

Difference between CoqGym and HOList:

1. Statefulness:

CoqGym is stateful. It always tackles the first goal at hand and handles backtracking using Undo command (SerAPI command). HOList is stateless. It applies tactics to goals that are provided to it.

1. API expression:

CoqGym uses a python class to handle proof environments while HOList uses a server-like interface, protocol buffer, that can be used for any programming language.

1. Data (theorems files and human proofs):

CoqGym transforms data as json files and loads them into the environment at initialization. Not sure with HOList

Analysis:

Advantage of Statefulness:

1. More like a typical gym environment for RL.

Advantage of Statelessness:

1. Can do batching more easily.
2. Can do Supervised Learning
3. Can still do RL, just need to carry a state around.

Advantage of python class environment:

1. More natural to code with.
2. Easier to implement. No need to learn protocol buffer

Advantage of protocol buffer:

1. Can be implemented and used on any programming language
2. Can be used for other ITP languages.

Isabelle API:

proof\_assistant.proto and CoqGymAPI.md are good enough. Everything there works the same for HOLight, Coq, and Isabelle.

1. Be familiar with 3 agents and choose which format of environment fits best for each. a) RL (e.g. PolicyGradient) b) Supervised learning (Like ASTact or gamepad) c) Differentiable agent (I will write a document summary or read Heling's Thesis/slides)