Software structure

Nuriya Nurgalieva, Simon Mathis, Lídia del Rio, and Renato Renner

Institute for Theoretical Physics, ETH Zürich, 8049 Zürich, Switzerland

February 14, 2022

A modular tool to test multi-agent quantum thought experiments. We introduce a software package, *QThought*, to run quantum mechanical thought experiments where agents are modeled as quantum systems and have to reason about each other's experimental results (Figure 1). Users can customize the following modules:

- **Protocol module:** specifies the experimental setting, including number of agents and other quantum systems, measurements and physical transformations carried out by different agents, and which chains of inferences we are interested in analysing.
- **Agent module:** specifies agents' physical memories and processors, defining how abstract reasoning is implemented as a physical process, e.g. as a quantum circuit.
- **Interpretation module:** defines the immediate inferences drawn from an experimental outcome, which depends on the interpretation of quantum theory applied by an agent.
- **Consistency module:** defines the axioms of abstract logic that determine how inferences are combined and how knowledge is propagated into complex reasoning; it also determines the subjects that agents are allowed to reason about.

Structure of the software repository. The open-source software package *QThought* is publicly available [1]. The software is written in ProjectQ [2], a quantum programming language based on Python. On the repository, one can find: installation instructions; instructions on how to run and customize the software; Jupyter notebooks with examples: simple test scenarios and Frauchiger-Renner thought experiment [3] for two different interpretations; explanations of the experimental settings and conclusions; descriptions of the physical modelling of rational agents as small quantum computers; and descriptions of the implementation of interpretations.

In the following, we give a quick recap of the founding methods of classes in the package.

Agents. An agent in our implementation is modeled as consisting of the following components:

1. a memory register (to store memory as a state);
2. a prediction register (to store prediction as a state);
3. an inference system (used to make an inference from the memory to the prediction system with the help of an inference table).

The *Agent* class is used to initialize an agent via giving the dimensions of the memory and prediction system; the list of class methods is given below.

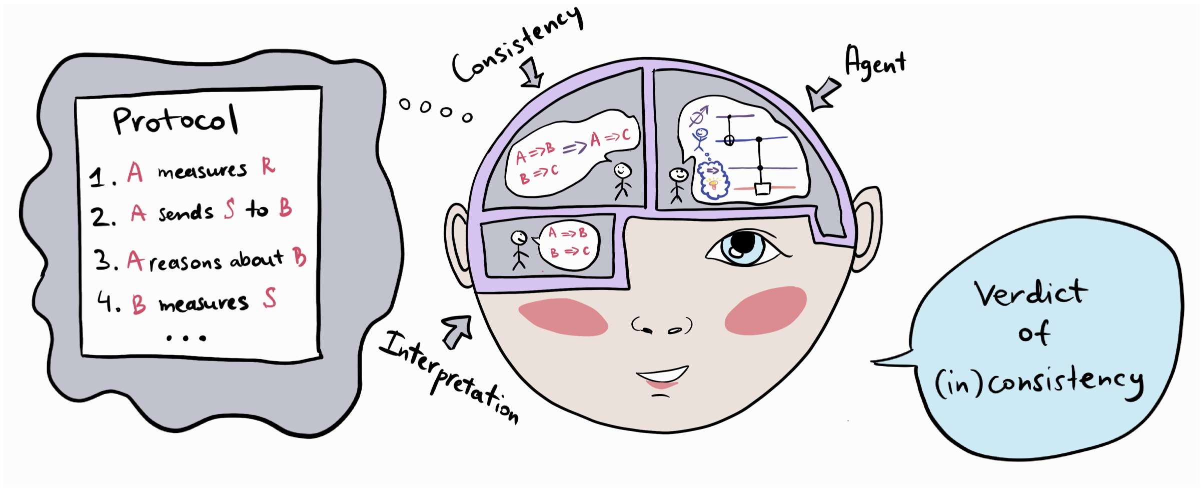

Figure 1: ***QThought* software structure.** Our software package allows us to model multi-agent scenarios where agents reason about each other's outcomes. It includes customisable modular components specifying: which inferences are made from experimental outcomes (*Interpretation* module); how those inferences are logically combined and propagated (*Consistency*); how agents' brains are physically modeled (*Agent*); and the experimental setting (*Protocol*). The program outputs the logical conclusions of each agent and whether they are compatible.

Methods of *Agent* class	
from_dim	used to initialize an agent via giving the dimensions of the memory and prediction system
__len__	returns the number of qubits of the quantum system
__getitem__	method to access the agent's qubits
memory	getter for the memory register of the agent
prediction	getter for the prediction register of the agent
inference_sys	getter for the inference system of the agent
all	getter for all registers that make up the agent combined
set_inference_table	initializes the agent's inference table with the given inference table
get_inference_table	getter for the agent's inference table
prep_inference	loads the agent's inference table into the inference system
make_inference	calls the inference operation, i.e. calls the circuit that copies the prediction state belonging to the state i of the memory into the prediction register
readout	reads out the memory and prediction registers and returns the results

Protocol. A protocol is a formal description of the experiment, and is represented as a *Protocol* class. A *ProtocolStep* class is used for filling out the *Protocol*.

A protocol instance has three attributes:

1. domain (list): required resources (qubits, quregs, agents) in the protocol step;

2. descr (str): describes the step in words ();

3. action (func): performs the action of the step.

A special *Requirements* class handles requirements for the protocol, and checks whether all necessary properties are met before running the protocol.

A detailed example of how one assembles protocol is given in a *simple example I* Jupyter notebook in the folder *simpleExamples*.

Utilities. The folder *utils* contains different functions which are used for the convenience of the program; for example, colouring the state vectors by individual subsystems, or a basic arithmetic library and calculation of an overlap between two states.

References

[1] Nurgalieva, N., Mathis, S., del Rio, L. & Renner, R. QThought (2021). URL `https://github.com/jangnur/qthought`.

[2] Steiger, D. S., Häner, T. & Troyer, M. ProjectQ: an open source software framework for quantum computing. *Quantum* **2**, 10–22331 (2018).

[3] Frauchiger, D. & Renner, R. Quantum theory cannot consistently describe the use of itself. *Nature Communications* **9**, 3711 (2018).