Software structure

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

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

September 27, 2019

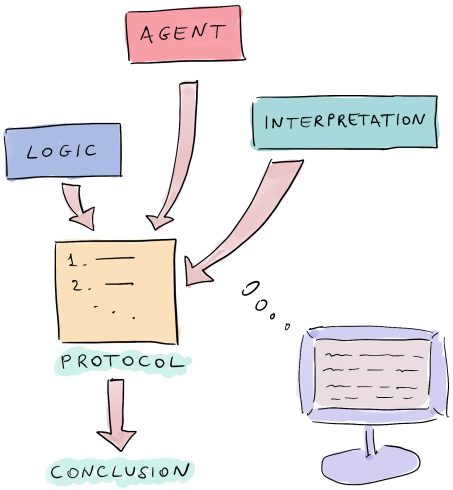

Figure 1: **High-level software structure.** The main elements of the framework are following: agent, logic, interpretation, which are called up in the protocol of a proposed experiment and are used to derive a conclusion about the setting.

Here we discuss the built-in features of the software, needed to put together any experiment at hand, namely, *Agent*, *Protocol* and *Requirements* classes.

1 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.

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

2 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*.

3 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