Trading Agent Example (display.py, agents.py, tagent.py):

Consider a household trading agent that monitors the price of some commodity (e.g., it checks online for special deals and for price increases for snacks or toilet paper) and how much the household has in stock. It must decide whether to order more and how much to order. The percepts are the price and the amount in stock. The command is the number of units the agent decides to order (which is zero if the agent does not order any). A percept trace specifies for each time point (e.g., each day) the price at that time and the amount in stock at that time. A command trace specifies how much the agent decides to order at each time point.

The action of actually buying depends on the command but may be different. For example, the agent could issue a command to buy 12 rolls of paper at a particular price. This does not mean that the agent actually buys 12 rolls because there could be communication problems, the store could have run out of paper, or the price could change between deciding to buy and actually buying. However, in this example we can see that the buy orders are all successfully executed, as the amount in stock went up immediately after the order to buy

A causal transduction specifies, for each time, how much of the commodity the agent should buy depending on the price history, the history of how much of the commodity is in stock (including the current price and amount in stock) and the past history of buying.
An example of a causal transduction is as follows: buy four dozen rolls if there are fewer than five dozen in stock and the price is less than 90% of the average price over the last 20 days; buy a dozen rolls if there are fewer than a dozen in stock; otherwise, do not buy any.

To implement the causal transduction, a controller must keep track of the rolling history of the prices for the previous 20 days. By keeping track of the average (average), it can update the average using

$$\text{average} := \text{average} + (\text{new} - \text{old})/20$$

where new is the new price and old is the oldest price remembered. It can then discard old. It must do something special for the first 20 days.
A simpler controller could, instead of remembering a rolling history in order to maintain the average, remember just a running estimate of the average and use that value as a surrogate for the oldest item. The belief state can then contain one real number (ave), with the state transition function

$$\text{ave} := \text{ave} + (\text{new} - \text{ave})/20.$$

This controller is much easier to implement and is not as sensitive to what happened exactly 20 time units ago. It does not actually compute the average, as it is biased towards recent data.