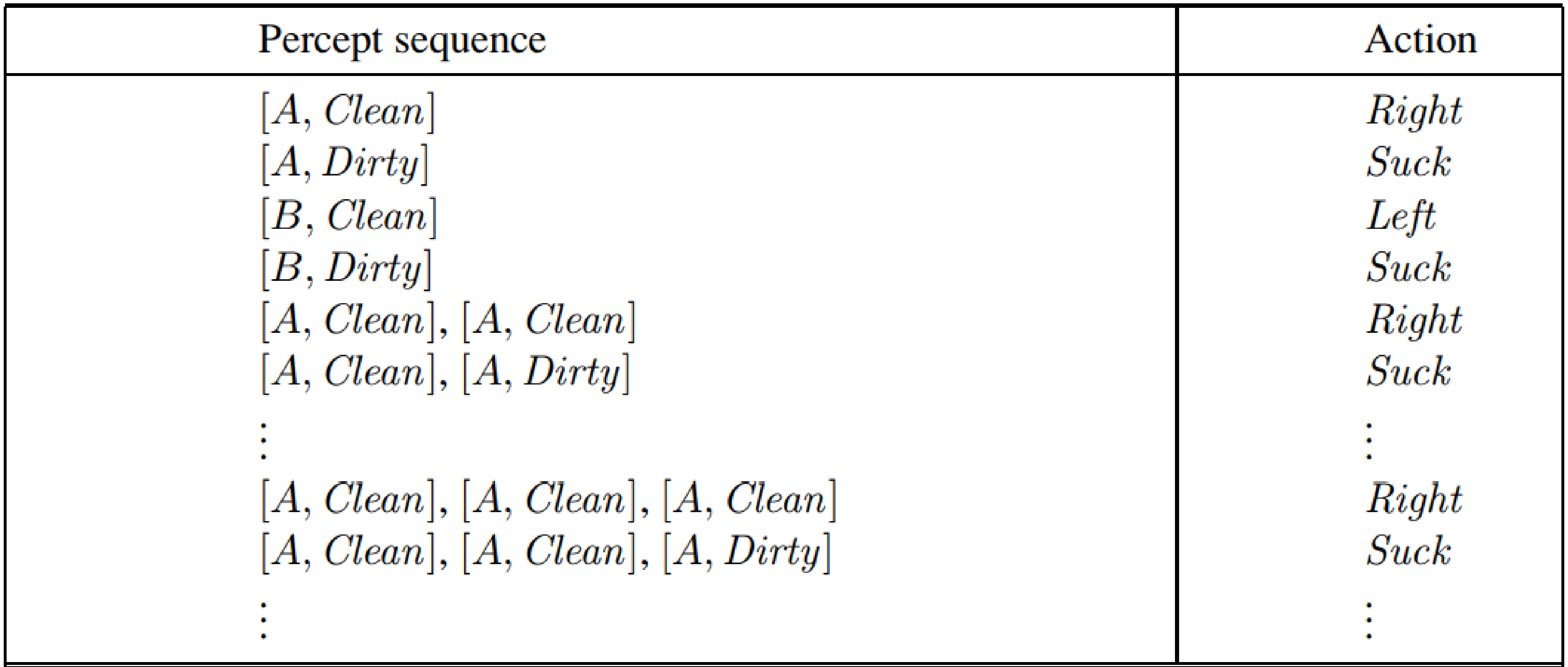
Let us examine the rationality of various vacuum-cleaner agent functions.

1. Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational under the assumptions listed on page



Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

2. Describe a rational agent function for the case in which each movement costs one point. Does the corresponding agent program require an internal state?

3. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? If not, why not?

It is easy to show the given agent function is equivalent to one that ignores the percept sequence and considers only the current percept. So there are really 4 possible states.

$[A, Clean] : Right$

$[A, Dirty] : Clean$

$[B, Clean] : Left$

$[B, Dirty] : Clean$

What constitutes rationality depends on the performance measure, agent's knowledge, action-space, agent's percept sequence. The performance measure measures the number of locations that are clean at any given time. When the two locations are dirty there's no other rational way to clean them than to do it in three steps (Clean, Left/Right, Clean). Similar is the case when one of the locations is dirty or no location is dirty.

Now let's consider the case where the performance measure penalizes movement. It no longer makes sense to move when we know the other location is clean. In this scenario, the agent could benefit from keeping track of which states are dirty/clean.

So the agent function also maintains a state that holds whether locations A and B are clean/dirty. This internal state is updated every timestep and when both the locations are clean, the agent no longer requires to take any action.

```
function UpdateState(state, percept) returns a state
    if percept.state is clean
        state[percept.location] = clean
    return state


function ReflexAgentWithState(percept) returns an action
    # the initial state is {A:dirty, B:dirty}
    state = UpdateState(state, percept)
    rule = RuleMatch(rules, state) # move only if other location is dirty
    return rule.action
```

When clean squares can become dirty over time and the geography of the environment is unknown, how would the agent decide the next action if the current percept says the location is clean? Should it move in anticipation of dirt in other locations? If yes, which locations should it move to?

This requires some model of the environment's geography which can be learned in finite time with enough exploration (assuming the geography doesn't change over time). One another thing the agent should learn is how different locations get dirty at any time. For example, if the locations in environment get dirty every x steps after cleaning then we want the agent to learn the value of x. It is also possible that different locations have different distributions that model the chances of getting dirty once cleaned. In this case, the agent should learn something more sophisticated like a probability distribution for each location indicating the chances of being dirty.