Incorrect Moves and Testable States

2 of May, 2017

How do we describe the invisible? Let's take a sequence: input, output, input, output ... Behind this sequence stands a world and the sequence of its internal states. We do not see the internal state of the world, but only a part of it. To describe that part which is invisible, we will use the concept of 'incorrect move' and its generalization 'testable state'. Thus, we will reduce the problem of partial observability to the problem of full observability.

Keywords: Artificial Intelligence, Machine Learning, Reinforcement Learning, Partial Observability, Diagnosability.

Introduction:

Our first task in the field of Reinforcement learning is to describe the current state of the world (or the current position of the game). When we see everything (full observability) this question does not arise because in this case the input fully describes the state of the world. More interesting is the case when we see only a part of the world (partial observability). Then we will add more coordinates to the input vector, and thus we get a new vector which already fully describes the state of the world.

To add these new coordinates we will first introduce the concept of 'incorrect move'. The idea that not all moves are correct is not new. Other authors suggest, for example, that you cannot spend more money than you have. That is, they assume that the output is limited by a parameter that changes over time. What's new in this article is that we will use this parameter to give further description of the state of the world.

If at a specific moment we know what we see and what moves are correct at that specific moment, we know a lot about the current state of the world, yet we do not know everything. When we generalize the concept of 'incorrect move', we will derive the new concept of 'testable state'. If we add to the input vector the values of all testable states, we will get an infinite-dimensional vector that fully describes the state of the world. That is, for every two distinct states there will be a testable state whose value in these two states is different.

Note: The closest to the term 'testable state' is the term 'diagnosability' (see [1, 2]). However, there are three significant differences between these two concepts:

1. Diagnosability corresponds to a semi-decidable testable state. This is so because with the testable state we have some experiment that, when it can be performed, will give us the value of a parameter. With 'diagnosability', whenever the experiment can be performed the parameter is necessarily true. When the parameter is not true, the experiment cannot be performed.

2. With 'diagnosability', we look at a parameter that changes its value only once (first, the failure event has not happened, and then it has already happened, that is, there is a single change from 0 to 1). With the testable state, we look at a parameter that can change its value many times, and so it is very important to us when exactly the parameter has changed its value, whereas in [1, 2] this question is not significant.

3. With ‘diagnosability’, we start with the parameter and look for experiments to diagnose this parameter. With the testable state it is the opposite. Testable state is a specific experiment itself. Later, a link is sought between the different testable states. For example, it can be determined that two testable states define the same parameter.

Incorrect moves and testable states are something that actually exists just like the input we get on each step. However, unlike the input, the value of testable states is not ready to derive. For example, is the door locked? To check this we need to push the handle and see whether the door will open, but we can do it only if we stand by the door. In other words, the check requires extra effort and it is not always possible (there are moments in which we can check and moments in which we can’t). The locked door can be considered as an example of both an incorrect move and of a testable state.

To describe the incorrect moves and testable states we will search for a theory that gives us this description. Of course, we may have many theories for a given testable state all of which will compete with each other over time.

What will the theory constitute? Statistics shows us that in specific situations a given testable state is true. Specific situations means a situation in which a conjunction is true. This conjunction may be associated only with the past, but may also be associated with the past and the future. For example, let’s say we’ve checked and we’ve seen that a door is locked, but is it the door that we are interested in? We may decide that it is precisely this door on the basis of what we have seen before checking, or maybe after checking, a posteriori, we’ve seen something which tells us that it is exactly this door.

Another generalization of ‘specific situations’ will be to allow dependencies with memory (except dependencies without memory). A dependency without memory is the situation in which specific events occur in the course of several consecutive steps (i.e. this is an ordinary conjunction). A dependency with memory is the situation in which specific events occur at specific moments (not consecutive), and in the periods between those moments specific events do not occur.

The theory may be a set of such implications and this set can be generalized as a finite-state machine. Let’s take an example where we are moving in a castle in which some of the doors are locked. We can have a rule of the following type: “If you see a white sofa and you turn right, then the door will be locked.” If we represent the map of the castle as a finite-state machine, we will see that if after the white sofa we turn right, we are at the door *X* and that this door is locked. If we know the finite-state machine, we can easily derive the corresponding rules. Unfortunately, we need the opposite – to derive a finite-state machine from the rules, and that’s a much more difficult task.

Let us now imagine a castle the doors of which are not permanently locked or unlocked but change following specific rules. Then our theory will suggest some sustainability of testable states. For example, if a door has been locked at a specific moment and shortly thereafter we check again, we assume that it will be locked again, especially if during this time no specific events have occurred (for example, to hear a click of door locks).

The next upgrade of the theory would be to assume that there is some kind of creature inside the castle, which unlocks and locks the doors in its sole discretion. Then we can predict whether a door is locked or unlocked predicting the behavior of that creature.

Once we've created one or several theories that predict a testable state, we will use these theories and gather statistics that will help us predict the future and to create new theories of other testable states. For example, in our case, if we've noticed that behind the locked door there is a princess, and a tiger behind the unlocked door, then based on the theory that tells us which door is locked, we can make a theory that tells us where the princesses are.

The article begins with a few definitions, then follows a specific example, the concept of 'incorrect move', and we say what the dependencies with and without memory are. We define the testable state and show that the incorrect move is its special case. We will say what an axiom is and how the axioms make theories. It remains to be said how from axioms we will make probable theories and how we will make more sophisticated theories based on a finite-state machines or group of creatures (agents) living in the same world and changing the testable states at their own discretion.

Definitions

Let's take a sequence of *output, input, output, input, …*, and the goal is to understand this sequence.

Of course, this sequence is not accidental. We can assume that we are playing a game and that's the sequence:
move, observation, move, observation ...
And our goal is to understand the rules of the game and what is the current position on the game board.

We might assume that we are in a world and then the sequence would be:
action, view, action, view ...
In this case, our goal is to understand the rules of this world and what is the current state of the world.

The description of the world is given by the functions *World* and *View*, and the following applies:

$s_{i+1}=World(s_i , a_{i+1})$
$v_i=View(s_i)$

Here, actions and observations (a_i and v_i) are vectors from scalars with dimensions n and m, respectively.

Our goal is to present the internal state (s_i) also as a vector of scalars, in which the first m coordinates will be exactly v_i. In other words, the function *View* will be the projective function that returns the first m coordinates of s_i.

We will call 'states' to all coordinates of s_i. We will call the first m ones 'visible states'. Other coordinates will be called 'testable states'.

The coordinates of the input and output will be called ‘signals’. These are functions of time that return a scalar. To the input and output signals we will add other signals as well, like the testable states; more precisely – the value of the testable state according to the relevant theory, because we will not know the exact value of these states, and will approximate them with some theories. For each finite-state machine we will add a signal whose value will be equal to the state in which the finite-state machine is situated at the moment *t*. Of course, if the machine is nondeterministic, the value of this signal may not be exact but approximate.

We will call the Boolean signals ‘events’. When the Boolean signal is truth, we will say that the event is happening.

Example

To make things clear, let’s take an example. Let’s imagine we are playing chess with an imaginary opponent without seeing the board (we are playing blind). We will not specify whether the imaginary opponent is human or a computer program.

Our move will be the following 4-dimensional vector:
X_1, Y_1, X_2, Y_2

The meaning is that we are moving the piece from the (X_1, Y_1) coordinates to the (X_2, Y_2) coordinates.

What we will see after each move is a 5-dimensional vector:
A_1, B_1, A_2, B_2, R

The first four coordinates of the input show us the counter-move of the imaginary opponent, and *R* shows us our immediate reward.

All scalars have values from 1 to 8, except *R*, whose value is in the *{-1, 0, 1, nothing}* set. The meaning of these values is as follows: *{loss, draw, win, the game is not over}*.

Incorrect move

Shall we allow the existence of incorrect moves?

Our first suggestion is to choose a world in which all moves are correct. In the example we took, we cannot move the bishop as a knight, so it is natural to assume that some of our moves are incorrect or impossible.

Our second suggestion is to have incorrect moves and when we play such a move to assume that the world penalizes us with a loss. So we will very quickly learn to avoid incorrect moves, but here we are denied the opportunity to study the world by checking which move is correct (like touching the walls in the darkness).

Our third suggestion is: When an incorrect move is made the state of the world to remain the same, and instead of 'loss' the immediate reward to be 'incorrect move' and all other coordinates at the input to return 'nothing'. This option is better, but thus we would unnecessarily prolong the history. (We will call 'history' the following sequence: $a_1, v_1, \ldots, a_{t-1}, v_{t-1}$, where t is the current time). Given that the state of the world remains the same, there is no need to increase the count of the steps.

The fourth suggestion is: When you play an incorrect move, you to be informed that the move is incorrect but without prolongation of the history. The new history will look like this: $u_1, a_1, v_1, \ldots, u_t, a_t, v_t$. Here u_i is the set of incorrect moves at the i-th step which we have tried and we know for sure that they are incorrect. Thus the history is not prolonged, yet the information that certain moves are incorrect is recorded in the history. Here we assume that the order in which we've tried the incorrect moves does not matter.

The fifth option, which we will discuss in this article, will be even more liberal. In the previous suggestion we have the opportunity to try whether a move is incorrect, but if it proves correct, we have already made this move. Now we will assume that we can try different moves as many as we want and we can get information whether it is correct or incorrect for each one of them. After that, we make a move, for which we already know that it is correct. We know because we've tried it. Here, the history is the same as with suggestion No. 5, except that u_i is a tuple of two sets, the first of which is from the tried incorrect moves, the second – from the tried correct moves.

There is a sixth option and it is for every step to get full information about which moves are correct and which are not. In other words, we get u_i with all the moves in it like they have been tested. However, we do not like this option because u_i may be too large, i.e. it may contain too much information. Moreover, we assume that after some training we will have a fairly accurate idea about which move is correct and which is incorrect and will not have to make many tests in order to understand this.

Dependencies without memory

We will assume that the current moment t is a very big number, and the history is too long and it is not worth remembering it all. Even if we remembered it, it would be foolish to process the whole of it on each step in order to choose our next move. So we will assume that instead of remembering the entire history we have collected some statistics and will determine our next move solely based on these statistics.

What will this statistics constitute? We will look at the conjunctions of literals. Here 'literal' means the equality between a signal for a given t and a particular value that this signal could receive. In the present example, with the game of chess, let us take for example the following conjunction:

$X_1(t)=5, Y_1(t)=2, X_2(t)=5, Y_2(t)=4, A_2(t)=5, B_2(t)=4$

This means that at the moment t we've moved a piece from E2 to E4 and the imaginary opponent has captured the piece that we've played.

Let's take another example:

$A_2(t\text{-}1)=5, B_2(t\text{-}1)=4, X_2(t)=5, Y_2(t)=4$

This means that imaginary opponent has played on E4 and we immediately, the very next move, have captured the piece that he has played.

We will assume that this conjunction are not scrambled and are ordered:
1. Older ones supersede new ones (i.e., *t-1* supersedes before *t*).
2. The output supersedes the input.
3. The coordinate *i* of the input (output) supersedes coordinate *i+1*.

We will assume that these conjunctions are stabilized, i.e. the time at the rightmost literal is *t*. What is the idea of stabilization? Conjunction is an event that is happening at the moment *t*. If we put *t+1* instead of *t* we get the same event that happens at *t+1*. Since this is the same event, possibly shifted in time, we will collect statistics only for one of all possible variants of the event. The value of literals at *t+1* is not yet known and we will therefore take the event whose value is already known and has become known exactly at the moment *t*.

For each of these conjunctions we will count how many times it has occurred (i.e. it has been truth).

Definition: We will call a move a conjunction, in which all literals are for the moment *t*, there are no literals from the input and all coordinates at the output are taking part.

Definition: A cumulative move will be the same as move, except that not all coordinates at the output will necessarily participate.

That is, the conjunction ‘move’ describes a particular move, while the cumulative move describes a set of moves.

Definition: We will call a conjunction, whose last literal is from the output, a conditional move. Such a conjunction can be represented as A & M, where A represents the condition, and M is a cumulative move. (Here we assume that we’ve divided it into A and M so that M is the maximum.)

For the conditional moves will not only count how many times they’ve been played, but also how many times they’ve been tried and how many times they have shown a correct or incorrect move when tried. That is, for every conditional move we will keep three counters:
1. How many times it has been played;
2. How many times it has been tried and it has been correct
3. How many times it has been tried and it has been incorrect.

The value of counter 2 will be greater or equal to the value of counter 1.

For all other conjunctions we will have only counter 1 (which will only remember how many times this conjunction was true). Here we will not count how many times there has been a correct move (because there has always been) and will not count how many times there have been an incorrect move (because it is information that is not worth the effort to collect).

Note: We will not consider the empty conjunction as a cumulative move. That is, we will not consider the set of all possible moves. We'll skip the signal which is truth when all possible moves are correct. We will skip this signal because the effort that we need to make to collect statistics about it is not worth it. Is there a case where all possible moves are incorrect? Yes, it can happen, but if it happens it will be only once. We will call this situation a cul-de-sac or premature death. In this situation, the history will end because there won't be a single possible correct move.

We will organize all conjunctions in a tree, where each conjunction will be a node of the tree. This will save memory, but mostly will save time because at each step we will not go around the entire tree, but only around that part of it where nodes are true conjunctions (i.e. true at this step).

When we go around the tree we will increase by one the counters which are on the nodes through which we pass. It is important to note that when we have several correct moves; they will be presented as a sub-tree of our tree. We need to go round this sub-tree and increase by one all counters for correct move in it. These nodes, which have several successors, should also be increased only by one. If we are not careful, we may pass through such a node several times and increase it by one each time, which would be a mistake. (What we've said about several correct moves applies correspondingly to several incorrect moves.)

Example: If at the current moment the incorrect moves are the cumulative ones: $X_1(t)=1$, $Y_1(t)=1$ and $X_1(t) = 1$, $Y_1(t) = 2$ (i.e., we cannot capture a piece from A1 and from A2) then we need to increase by one the counters of these cumulative moves (counter 3 – for the case when in this cumulative move there has been an incorrect move). Counter 3 for the cumulative move $X_1(t)=1$ must also be increased by one because there has been one moment in which this cumulative move contained an incorrect move, regardless of the fact that in this case it contains not one but two incorrect moves.

With the use of this tree we will collect statistics on how many times each of the observed conjunctions has been a truth. Of course, all of conjunctions are countless, so we will observe only some of them (which are shorter and include fewer steps). We'll assume that if we are observing a conjunction, we are observing all its subsets.

However, the information about how many times a conjunction was a truth is not sufficient. We also want to know when it was a truth for the last time. This can easily be achieved by adding another memory cell to the counter (to all three counters) we've already attributed to this conjunction. Every time we increase counter 1, we will record the value of the counter of time in that memory cell.

Let's say we want to know not only the last time when the conjunction was a truth but the last 100 times when it was a truth. This can easily be achieved with the expense of memory, but without the expense of computer time. Instead of an additional cell we will attribute a hundred cells. In which cell will we record the value of the counter of time? We'll take the value of counter 1 by module 100 and this will give us the number of the cell.

Thus we will keep some very strict memories of the last 100 steps, and even of more than 100 steps, because we will know what conjunctions were well back in the past for these conjunctions, which are rarely a truth.

That will not be enough and we want to have more information about the time before last hundred successes of the event. This information will not be entirely accurate, but it will be approximate. For this purpose we will choose 100 important points that will divide the history to 101 intervals. We will count how many times the event occurred in each of these 101 intervals. More precisely we will count how many times it happened from the beginning (from the moment of birth) to the end of each of these intervals, but the difference between two such numbers will give us how many times it occurred between any two important points.

Those hundred important points will not be static and will change over time. The idea is those are densely when close to the current moment, and less frequent back in the past. Here is a sample algorithm for changing the important points:

We put an important point at every 10 steps. When the 100 points finish we begin rarefaction of the important points, one in one, starting from the beginning to the current moment. When we finish the first rarefaction we start the second one, again from the beginning and so on.

It is not very smart to put important points at regular intervals. It would be smarter to specify some important events and put important points when such an event occurs. For example, if you are trying to find a finite-state machine whose transitions are certain events, it would be smart to put important points when these events (transitions) occur. This would help us to gather statistics on this finite-state machine and to build it for more easily.

How will we organize the counting? This will again be at the expense of some memory, but without the expense of computer time. For each conjunction we will set aside another 100 memory cells for the one hundred important points. Thus, the cells for each conjunction become 201 (203) total.

Let's make an auxiliary table. It will consist of 100 rows and two columns. The first column will be the times of the one hundred important points (sorted) and the second will be the numbers of points (which are between 0 and 99). This table will help us to calculate how many times a conjunction between two important points has been a truth. We will choose from the table the two moments that define our interval of interest. We take from the table the numbers of the respective important points. We check the last time the conjunction has been a truth and if the two are before this point, we take the difference. If both are after, the answer is zero. If the first moment is before, and the second after, then we take the difference between the counter 1 and the cell of the first moment.

This auxiliary table will be used in counting as well. Here's how the algorithm of counting will look like:

When a conjunction is a truth, we take counter 1 by module 100 and this is the address of the cell which remembers the last time this conjunction has been a truth. We take this number. In the next cell (by module 100) we record the current time. We look in the auxiliary table from the bottom to the top and we go until the time of creation of the important point is after the time we remember (the last time this conjunction has been a truth). We take the number of each such (new) important point and write down the value of counter 1 in the cell which address is this number. When finished, we increment counter 1 and we are ready.

Testable states

A testable state is the result of an experiment. Here are two examples:

The door is locked
If I push the handle ⇒ the door will open.

The stove is hot
If I touch the stove ⇒ I will get burned.

Here we have the testable state and the experiment, which must be made in order to obtain the value of the testable state. The result of the experiment can be 'Yes' or 'No'.

From the above two examples you might get the impression that the experiment consists of any actions that we need to do, but it is not so. The experiment could be something that happens without our intervention. Here is an example:

The roof is damaged
If it rains ⇒ the roof will leak.

In this example the testable state is checked when it rains, and this is something that does not depend on our actions.

Definition: A testable state will consist of two statements. We will call the first one 'condition' and the second one 'conclusion'. Below we will give additional requirements to the condition and the conclusion of a testable state.

Definition: We say that a testable state is a special case of another one if they have the same conclusion, and if the condition of the second is a special case of the condition of the first.

Note: We can think of the testable state as an implication, although this is not quite accurate because its possible values are three (true, false and undefined – when the condition is not true). If we imagine this implication as a disjunction, a special case is the disjunction that has more literals. Accordingly, the conditions will be conjunctions, and a special case is the conjunction that has fewer literals.

Let's define incorrect move.

Definition: An incorrect move is testable state whose condition is a conjunction in which all literals are from the output and are for the moment $t+1$ and whose conclusion would be "the move is incorrect."

That is, the incorrect move is testable state, whose condition is a cumulative move (at the moment $t+1$).

To complete the definition of testable state we will first define an immediately testable state is. This is testable state that, if it can be tested, the test will be carried out at the next step.

Definition: An immediately testable state is one of the following:

1. An incorrect move.
2. A testable state, whose condition is a conjunction, in which all literals are for the moment *t+1*, and the conclusion would be a literal from the input, also for the moment *t+1*.

Here two questions arise. Why the conclusion is only one literal and why the literal is from the input? The answer to the first question will be: If the conclusion is the conjunction of two literals, then we can present this testable state by four other testable states in which the conclusion is only one literal. If this testable state is of the type $A \Rightarrow B_1 \& B_2$, this testable state is a truth just when the testable states $A \Rightarrow B_1$ and $A \Rightarrow B_2$ are a truth. This state is a lie just when the testable states $A \& B_1 \Rightarrow \neg B_2$ and $A \& B_2 \Rightarrow \neg B_1$ are a truth. (Here the negation with Boolean signals is one literal, and with non-Boolean ones – a disjunction of several literals.)

As for the second question: We want to describe the world, not to describe our behaviour. That in certain circumstances we have always played something (or have never played this thing) is not a description of the world, and a description of our behaviour. We may never have played it because it's an incorrect move or we could have always played it, because this is the only correct move. This latter is already a description of the state of the world, but this information is given by the term 'incorrect move'.

Having defined an immediately testable state we are to get a definition of the term 'testable state' by adding something to the condition. We will add precondition or postcondition or both. A precondition will be a conjunction of literals which are for moments before *t+1*, and a postcondition will be a conjunction of literals which are for moments after *t+1*.

Here, one part of the conjunction is associated with the past (before *t+1*), and the other part is associated with the future (after *t*). The testable state is a prediction for the result of carrying out an experiment that has yet to be held. So a testable state must fully involve the future. Otherwise, we get something that is partially known and it can not be verified (for the known part is a lie) or will be a testable state, but some other testable state (the first will be a special case of the second) because the known part will be a truth and the new testable state will depend only on the part that is still unknown.

Therefore we will correct the definition of a testable state by shifting the entire conjunction to the right, so it all goes into the future (we add a *k* to the time of all literals). We need to shift the conjunction to the right until the most left literal is for the moment *t+1*.

If we shift further to the right (e.g. by one more step), we will get another testable state, but it will be a prediction of more distant future. So this new testable state will be a truth if the old testable state is a truth at step *t+1* whatever happened at step *t+1*. That is, the new testable state will be a more common case of the old taken for step *t+1*. This is so because the new one can be obtained from the old one taken for step *t+1* by adding what happened at step *t* (i.e. to add a conjunction describing completely the output at step *t*). That is, we quite naturally get that the testable state that looks ahead to the future, is a more general case of the one that looks closer.

In the definition of a testable state we can assume that the precondition and the postcondition are not conjunctions but cumulative conjunctions. That is, the precondition and the postcondition can have memory and apply for a longer period of time (and not just for a few steps).

Note that the number of immediately testable states is final (if the input and output signals are finite functions). By adding preconditions and postconditions we get an infinite number of testable states, which is natural, because we can not describe an infinite world with finitely many parameters.

When do we test?

We want for each testable state to assign a signal. That is, we look for a time function that is related to the result of the experiment. The question arises if the experiment is conducted over a period of time, when exactly the testable state signal takes the experimental result as its value.

If the experiment was limited to just one step, it would be easy, but it could take several steps. If the condition of the testable state is a common conjuncture, then we know how many steps this experiment takes, but if it is a cumulative conjunction, we do not know even that.

We will look at several options:

1. Let's assume the signal receives the value at the moment the experiment ends. That is, at the moments in which the experiment ends the signal receives a value and it is equal to the result of the experiment, and at the rest of the moments the signal is undefined. Under this definition, the value of the signal depends entirely on the past (history) and does not depend on the future (the current state) at all.

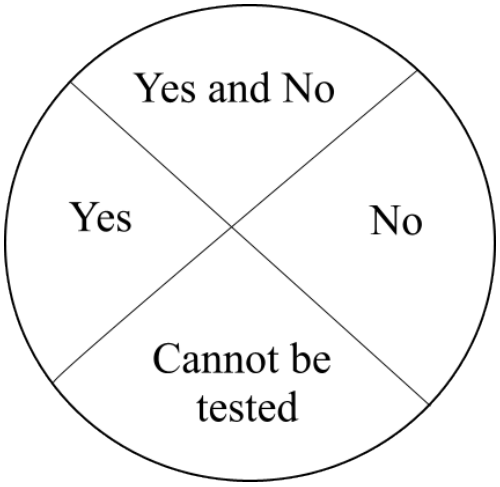
2. Let's assume the value is obtained at the moment when the testable state conclusion is tested. At this moment, the value is already clear, but the postcondition has not yet been verified, and it may be true or false, depending on what we will do (what are our future moves). We will assume that the signal will have value at this moment if the postcondition can be fulfilled (assuming we play it so to fulfill it). In this definition, the value of the signal depends on both the past (history) and the future (the current state).

3. Let's choose the moment immediately before the experiment starts. At this point, we still do not know what will be the result of the experiment, and whether it can be conducted. The value of the signal will be the forecast of the result of the experiment. This will be the definition we will choose. The good thing is that this definition depends only on the future (on the current state).

4. Let's choose a moment before the beginning of the experiment, but not immediately before. Then the signal value should still be a forecast, but this will be a more distant forecast. The possible developments seen from a distance are more. For example, if you see from a distance that the experiment cannot be performed, then from a closer distance you will see the same, but not vice versa. That is, in this way we would get another, different signal that corresponds to a more generally testable state.

The testable state is something real

The testable state is something quite real, as the input we get at every step is real. We can look at the value of the testable state as a function that depends only on the current state of the world. This function takes one of four possible values:



We get the value ‘Cannot be tested’ when the experiment is not possible. When does this happen? Let’s look at all the possible developments (all the moves we could play starting from the current moment *t*). If the experiment cannot be performed with any of these developments, then for moment *t* we have: ‘Cannot be tested’. It should be able to perform the experiment immediately, i.e. its beginning must be at the moment *t* and not later. If the experiment cannot be performed, it means that the precondition or the postcondition is false, or the condition of the respective immediately testable state is false.

Accordingly, ‘Yes’ will be if it can be tested, and always when it is possible to test it the result is ‘Yes’. It will be similar for ‘No’. The ‘Yes and No’ will be in all other cases.

When is the value ‘Yes and No’? This is possible in a loose experiment. This can also happen if the function *World* is not deterministic.

What does a loose experiment mean? This is when only some of the output signals are determined by the condition of the testable state. When all output signals are determined, then we will know which is exactly the move we have played, and then we will call the experiment tight.

Example:
If I touch the stove ⇒ I will get burned.
This experiment is loose because we have not said whether we will touch with a glove or without a glove.

If I touch the stove without a glove ⇒ I will get burned.
This experiment is tighter because the move has more detailed description.

Note: Here, if the precondition and the postcondition are elementary conjunctions, then the number of all possible developments will be finite, because then they will depend only a few steps before and after the test. If these are cumulative conjunctions, then all possible developments can be infinite, but a testable state still will be a well-defined function, although this function may not be computable. (We mean could we calculate it if we had the function *World* as a subroutine. Of course, we do not have the function *World*, so this remark is purely theoretical.)

Theories

The testable state is not ready to be obtained. As we have said, it is something real, but we cannot directly measure it, but we need to approximate it by theory.

The theory will be a function that will take the history as an argument and return a signal. This signal will have three possible values: *{Yes, No, I don't know}*.

The purpose of the theory will be to describe the signal of the testable state. We will say that the theory is correct if, whenever it says 'Yes' the signal of the testable state has one of these two values: *{Yes, Cannot be tested}*. (The same shall also apply for 'No'.)

We will say that the theory is complete if, whenever the signal of the testable state is 'Yes' the theory says 'Yes'. (The same shall also apply for 'No'.)

We will say that the theory is super complete if it is complete and if it makes a prediction of all the moments when the signal is 'Cannot be tested'. That is, for any such moment, the theory is to say 'Yes' or 'No'.

The theories that we will use will not be correct nor will they be complete, but we will strive to make them as correct and as complete as possible.

Every theory gives us some predictions about the value of the testable state, and this prediction has some degree of confidence. That is, for any moment, the theory will return two values: a prediction and a degree of confidence of this prediction.

Axioms

Definition: We will call a positive axiom a testable state that, whenever it was tested, it was true and that has been tested a sufficient number of times.

Similarly, we will define a negative axiom.

What does "whenever it was tested" mean? It means that this is so in the history we have accumulated. There is no guarantee that this will be the case with all possible histories, and there is no guarantee for all possible extensions of the current history. However, if the axiom is checked for a sufficient number of times, then with a high degree of confidence we can assume that this will always be the case.

What does "a sufficient number of times" mean? Let it be 10 times. If something has happened only once, it could have happened by accident, if it has happened 10 times, then it increases the degree of confidence, but of course there is no guarantee that the 11^{th} time the opposite will not happen.

Instead of counting to 10, it is better to want to have so many experiments that the expectation that the conclusion of the condition does **not** happen to be at least 10 times. If the conclusion has probability near zero, then to test 10 times will be enough. If the conclusion has probability about 50%, then we will have to test at least 20 times, and so on.

The conclusion is a specific literal and its probability can be taken from statistics by seeing how many times this literal was true and divide it to the counter of the steps. It is better not to take the probability of the conclusion but the probability of the small testable state (the one this axiom will describe). This value will still be taken from the statistics. How many times this small testable state has been tested and how many times it was true when tested.

Theories of axioms

What are we going to use the axioms for? If a testable state is a positive axiom, there is nothing to describe it. Its theory is the one that always says 'Yes'. Interestingly, with the help of axioms we will describe the smaller testable states (those to which the particular axiom is a special case of).

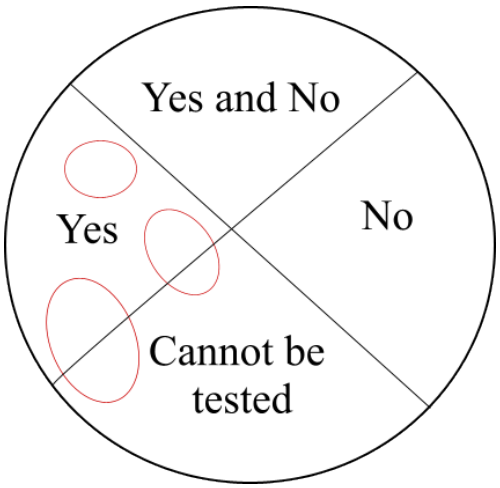
Let's take a testable state. From the statistics we will find axioms that are a special case of this testable state, and we will use these axioms to describe this testable state.

Definition: A prerequisite for an axiom will be the difference between the axiom and the small testable state it describes. (That is, the prerequisite depends on what we describe with this axiom.)

From each positive axiom we will make a theory that describes the small testable state and says it is true always under certain circumstances. The certain circumstances are when the axiom's prerequisite is true.

Let's say we want to predict the value of a testable state for the moment *t* based on the information we have at moment *t*. We will take the set of all histories with a length *t*. Each of these histories takes us to a certain internal state (several histories can take us to the same internal state). We divide this set of histories into 4 as we've divided the set of internal states into 4 (Fig. 1) by placing the history in the corresponding quarter based on what internal state it takes us to.

We have taken three axioms whose prerequisites are before *t* (that is, if we shift the axiom to the left, so that the small axiom starts from the moment *t*, then the whole prerequisite must be before *t*). For each of these three axioms, we draw a red circle surrounding the histories in which the prerequisite of the corresponding axiom is true.



Note: An internal state may correspond to two histories and one could be in the red circle and the other – outside it, but both will surely be in the same quarter.

We will assume that the axioms we have found through the statistics are correct. That is, we assume that their red circles are entirely in the 'Yes' and 'Cannot be checked' quarters. Of course, this assumption may not be true, and the red circle of any of these axioms may contain a state that could generate a 'No' result.

The red circle may be completely in the 'Yes' quarter, but it may also be partially in it. Can the red circle be completely in the 'Cannot be checked' quarter? We've checked this axiom in our history many times. So there is at least one internal state that could generate a 'Yes' result. The problem is that, although this state is reachable, there is no guarantee that it can be reached with a history of length *t*. For this reason, we change the description of Figure 2 and thus all stories will be included there, not just those with a length *t*.

We would like all histories to end at the moment *t*, which would be difficult if they are of different lengths. Here, the numbering of the steps in the history is largely conditional, and so we can align them to the right instead of aligning them to the left. That is, instead of the numbering always to start from 1 we will make it to end at *t*. So we will start the numbering of the shorter histories from a larger number, and the longer ones – from a negative number. (We are interested in how the history ends, not how it begins, and so we can safely number it backwards.)

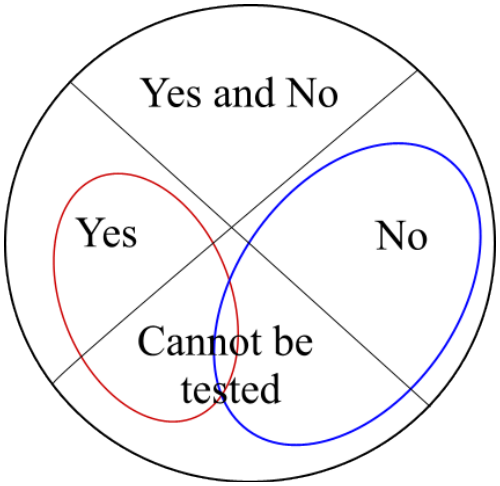
Once we have made this change, all histories and all reachable states are now included in Figure 2. Now we can say that the red circle can not be entirely in the quarter of 'Cannot be checked'.

This was for the moment *t*, based on the information we have at the moment *t*. We will want to be able to predict the testable state both at earlier and later moments. Very often the late prediction is useless like after death, the doctor, but it will still be helpful in collecting statistics. In addition, sometimes the late prediction may be useful and give us important information. (We are talking about a testable state. Because the moment is over, it does not mean that we already know what its value was at that moment. For example, we understand that at lunchtime the stove was hot, which means that the children have had lunch. This is useful information obtained from the value of a testable state at a past moment.)

Now we will make Figure 2 so that it depicts the axioms that predict the value of the testable state for the moment *t*, based on the information we have at the moment *t+k*. We had taken all the histories that ended at the moment *t*, and for each such history we had assigned the state to which it brought us at the moment *t*. We now want to take all the histories that end at the moment *t+k*, and for each such history we have assigned the state to which it has brought us at the moment *t*. What do we do when *k* is negative? Then this history does not reach the moment *t*. For this purpose we will take all the histories that finish at the moment *max (t, t+k)*. When *k* is negative, the last few steps of the history will not be used to determine if the prerequisite of the axiom is valid, because these few steps will still be unknown.

We received a simple picture with a very complicated description of what is depicted in the picture. Despite the complexity of this description, the algorithm that follows from this picture is quite simple. We take all positive axioms that describe a given testable state. We unite the red circles of all these axioms. We get a red circle that describes the histories for which our theory

says that the testable state is true at the moment *t*. We act with the negative axioms in a similar way and we get a blue circle. Figure 3 shows how our theory looks like:



Note that the red and blue circles can cross. For the cases where the theory tells us both 'Yes' and 'No', we will assume that it says 'I do not know'.

What is the algorithm? At each step, we check which axioms are coming out (that is, their prerequisite is already known and true). If, until now, our theory has said, 'I do not know because I have no information', then, after the axiom, the theory already knows and predicts something. If our theory already predicts 'Yes', then the new axiom, if positive, will only confirm this prediction, and if it is negative, the prediction will become: 'I do not know because I have too much information.'

Note that when the axioms are ordinary conjunctions, then an axiom gives us a prediction for a moment, but when the conjunctions are cumulative, an axiom gives us a prediction for a whole interval of time.

Conclusion

In the field of AI our first task is to say what we are looking for, and the second task is to find the thing we are looking for. This article is entirely devoted to the second task.

Articles [3-5], which are devoted to the first task, tell us that the thing we are looking for is an intelligent program that proves its intelligence by demonstrating it can understand an unknown world and cope in it to a sufficient degree.

The key element in this article is the understanding of the world itself, and more precisely – the understanding of the state of the world. We argue that if we understand the state of the world, we understand the world itself. Formally speaking, to understand the state of the world is like reducing the problem for Reinforcement learning with partial observability to the problem for Reinforcement learning with full observability. Not accidentally, almost all articles on Reinforcement learning deal with the case of full observability. This is because it is the easiest case. The only problem in this case is to overcome the combinatorial explosion. Of course, combinatorial explosion itself is a big enough problem because we get algorithms that would

work if you have an endlessly fast computer and indefinitely long time for training (long in terms of the number of steps). Such algorithms are completely useless in practice and their value is only theoretical.

In this article we presented the state of the world as infinite-dimensional vector of the values of all testable states. This seems enough to understand the state of the world, but we need a finite description and we would like this description to be as simple as possible. For this purpose, we've made three additional steps. A model of the world was introduced as a finite-state machine. Each such machine describes an infinite number of testable states and this is a very simple description which is easy to operate.

The second step was the assumption that testable states are inert and change only if specific events occur. That is, if between two checks, none of these events has occurred, we can assume that the value of the testable state has not changed.

The third step was the introduction of agents which under certain conditions may alter the values of testable states. This step is particularly important because the agent hides in itself much of the complexity of the world and thus the world becomes much simpler and more understandable. We will not describe the agent as a system of formal rules. Instead, we will approximate it with assumptions such as that it is our ally and tries to help us or that it is an opponent and tries to counteract.

Without these three steps the understanding of a more complex world would be completely impossible. Formally speaking, testable states only would be sufficient to understand the state of the world. Even testable states which conditions are simple conjunction dependent only on the last few steps of the past are sufficient. However, without the additional generalizations made, we would face an enormous combinatorial explosion.

Acknowledgments

I would like to thank my colleague, Dimitar Guelev, who introduced me to the concept of 'Diagnosability' and the related literature.

References

[1] Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control, 40(9):1555–1575, sep 1995.*

[2] Xiaowei Huang. Diagnosability in Concurrent Probabilistic Systems. *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

[3] Dobrev D. AI – What is this. *PC Magazine – Bulgaria, November'2000, pp.12-13.* http://www.dobrev.com/AI/definition.html

[4] Dobrev D. Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.* http://www.dobrev.com/AI/

[5] Dobrev D. Comparison between the two definitions of AI. *arXiv:1302.0216, January, 2013.* http://www.dobrev.com/AI/