Description of the Hidden State of the World

Dimiter Dobrev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
d@dobrev.com

The goal of AI is to predict the future and use this prediction as a basis for choosing its further course of action. AI tries to understand how the world works which means that it should find a model of the world. That model consists of internal states and the function that drives transitions from one internal state to another. AI will need that model in order to predict the next observation, i.e. in order to predict the future.

For AI to gain self-awareness, it must find the answer to the questions "Where am I?" and "What is going on?". The answer to these questions is hidden in the internal state of the world. An AI which does not endeavor to understand the world is weak AI. The way to creating a strong AI goes through the description of the internal state of the world.

If we are to create Artificial General Intelligence (AGI), it would not be sufficient just to learn how to describe the internal state of the world. We also need to move from single-step to multi-step reasoning. This means that we should be able to start from the current state of the world and mentally take several steps into the future, and thereby select the course of action that works best for us.

Keywords: Artificial General Intelligence, Partial Observability, Language for description of worlds, Hidden state of the world, Event-Driven model.

Introduction

We are interested in the answer to the question "What is going on?". That is, we are interested in the current state of the world. By watching TV news we learn what the current state of the world is. What happened 100 years ago may also be interesting, but the current developments are more important.

The state of the world can be divided in two parts: a visible part and a hidden part. What we have for sure is the current observation, which is the visible part. But, is there something hidden which is not visible? Things would become very simple if we assume that nothing is hidden out there. This approach is known as *Full Observability*, and is the prevalent approach used in AI.

In this paper we adopt the less developed approach of *Partial Observability* which assumes that the state includes something hidden as well. This is the more interesting approach because the description of the hidden part can be likened to fantasy. When we have to imagine something invisible, we resort to our imagination.

The title of this paper, namely *Description of the Hidden State*, is meant to emphasize that we do assume the existence of some invisible part. In fact, we will try to describe the entire state of the world without dividing it in visible and hidden parts.

Note: The terms "environment" or "nature" are sometimes used interchangeably with "world".

We will assume that the world has its own model which describes it. Let this be the world's *natural model*. This will be an abstraction which we will aim to reach but will be unable to reach because the natural model cannot be uniquely defined. There are infinitely many models – each one corresponding to our lived experiences – and the natural model is only one of them. The only assumption we can afford about the natural model is that it is simplified to the maximum extent possible. This assumption is known as Occam's razor, and it is a very important assumption because without it we would not be able to find anything.

Thus, we will not attempt to find the natural model of the world (the g function). Instead, we will try to find our idea of world (function g'). The way we perceive the world is a particular model, however, we will not operate with this model either, because with this model the state of the world is not uniquely defined. The model we will operate with will be called *description of the state of the world* (function h). These will be textual descriptions written in some language for description of worlds. At each moment of time we will have one and only one description of the state of the world. Although the so-obtained description may describe multiple internal states which exist in our idea of the world, the description we are going to use will package this set of internal states as a single internal state in the describing .

Predicting the future

Let us have a sequence of observations and actions:

$$o_0, \quad a_0, \quad o_1, \quad a_1, \quad \dots \quad , \quad o_t, \quad a_t$$

We will call this sequence *history* (or *current history*). Our aim will be to predict o_{t+1} (the next observation in this history).

We will assume that the world changes its internal state after each observation. Let after observation o_i the world be in state s_i. In order to predict the future, we will need the last internal state of the world s_t and the function g.

$$g(s_i, a_i) = \langle o_{i+1}, s_{i+1} \rangle$$

Note: We will determine the last internal state s_t on the basis of the current history and the function g will be determined on the basis of the entire lived experience. What is the difference between history and lived experience? If we had only one history, there would be no difference at all, however, if there are more histories then lived experience consists of all these histories.

In addition to the next observation, the g function also predicts the next internal state of the world. We will need the next internal state in order to move forward in the future and predict several steps ahead.

Accordingly, the g function produces two predictions. It makes perfect sense to split these two predictions in two separate functions: Function g_1 will predict the next observation, while function g_2 will take the already known o_{i+1} and will predict the next state of the world.

$$g_1(s_i, a_i) = o_{i+1}$$
$$g_2(s_i, a_i, o_{i+1}) = s_{i+1}$$

We can assume that the g function is single-valued (deterministic) and therefore predicts exactly what the next observation and accordingly what the next state of the world will be. Certainly such

a deterministic function exists, however, we will have hard time if we try to find it. That is why we will assume that the g function is multi-valued (nondeterministic) and produces several versions (scenarios) of the future. Even more so, when predicting the next observation we will assume that the prediction comes with probabilities. In other words, our assumption is that instead of a particular observation, the function g_1 returns some *believe* (all possible observations together with their respective probabilities).

Levels of Uncertainty

The of probability theory typically deals with uncertainty only at its first level (in this case we do not know whether an experiment will be successful but are hopeful that there is some exact probability of success). The assumption in this case is that the experiment is independent (i.e. there are no other factors which can be posed as additional parameters).

However, in our case we cannot assume that the experiment is independent. Accordingly, our experiment comes with a second level of uncertainty – in addition to being uncertain about the success of the experiment, we have no idea what the exact probability of success may be. All we know is that there is some interval in which the exact probability resides.

We will be looking for a model which has a second level of uncertainty. Furthermore, we will assume that the natural model has a second level of uncertainty as well. As we said, we can assume that the g function is single-valued (deterministic). Now we will let randomness in the model and say that such randomness cannot be predicted. Moreover, we will assume that the probability of such randomness varies in certain limits and this variation cannot be predicted, either.

This means that the g_1 function will return *believe* in probability intervals. A detailed study of these intervals in provided in [4].

Hidden information

As regards the prediction of the next state of the world, it would be beneficial that at least the g_2 function is a deterministic one. A deterministic g_2 will mean that there is no hidden information in the world.

Note: This paper deals with hidden information and hidden state. Hidden information is indeed hidden state, but the opposite is not true. A hidden state is information which cannot be derived from the last observation, while hidden information is something that cannot be derived from the entire history until the present moment.

In [4] we considered a world in which someone has sent us a letter and we do not know what the letter says before we open it. Therefore, in this world the content of the letter is hidden information. We can assert that once the letter is sent the g_2 function changes the state of the world in a nondeterministic way and produces two different worlds. In one of these worlds the letter will bring to us some good news and in the other world we should be prepared to read some bad news.

Any model which contains hidden information can be processed to an equivalent model which does not contain any hidden information. We can hypothesize that in the model there may be

"undetermined" states which will get determined later. Nonetheless, it is not a good idea to discard hidden information because this would complicate the model. For example, we do not assume that the content of the letter is "undetermined" before we read it and then gets determined at the time of reading. Therefore we will assert that our idea of the world includes hidden information but in the description of the world all hidden information is packaged in one description. The fact that a letter has been sent to us does not create different descriptions. Instead, the sending of the letter results in a single description which predicts different scenarios. In other words, all hidden information is described in the description of the world but does not create non-determinism (in the form of various possible descriptions).

Now we will address the following question: Can we stay only within the confines of models in which everything is visible (i.e. the observation says everything about the state of the world)?

Do we need memory?

This question seems rhetorical – natural intelligence (i.e. the human being) needs memory, so the artificial human being (AGI) should also have memory.

Although the answer to this question seems obvious, most authors nowadays consider AI as a memoryless device (i.e. as a function which depends only on the current observation and does not depend on what has happened until present).

This is the *Full Observability* approach which assumes the absence of any hidden information. Thus, everything that matters is visible in the observation. If there is anything hidden, it is unimportant and does not affect future development.

If we see everything important, do we need any memory? Of course not. We only need memory to remember things that we have seen in the past but are no longer before our eyes at present. (Memory does not determine everything. On one hand there are things we have not seen but on the other hand there may be important things which we have seen and memorized.)

In case we can see everything, can the world change its internal state? In other words, will the world have its own memory (the internal state of the world is the memory of the world). The answer is that the world will be able to change its internal state, but its state will be visible in the observation. If we are to learn from our mistakes (or from our own lived experience) then the new state of the world will be important. In that case we would divide the states of the world in better states and worse states, and strive to bring the world to a better state. With the *Full Observability* approach we assume that instead of learning from our own lived experience we rely on the experience of some expert who knows exactly what needs to be done. Therefore, we will not be aiming to bring the world to a better state, but will only try to replicate the actions of that expert because he knows exactly what needs to be done.

That is, in *Full Observability* we would not be interested in the internal state of the world, or to put it more simply, we would assume that the world has one and only one internal state (i.e. it does not have any memory).

With *Full Observability* we try to replicate the behavior of our expert based on his lived experience. The expert's lived experience will be in the form of a set of 2-tuples $\{observation, action\}$. Why a set and not a list? Because the order of the tuples does not matter.

The form of the expert's lived experience is:

$$\{ \langle o_i, a_i \rangle \mid i \in I \}$$

Let the behavior of the expert be described by a function f.

$$\forall i \in I \quad f(o_i) = a_i$$

We will try to find the function f' which is an approximation of f. That is, we will look for the simplest function such that:

$$\forall i \in I \quad f'(o_i) = a_i$$

The above approximation is usually sought in the set of neural networks. The neural networks method yields astonishing results. By this method we will end up with a very smart program, but this program will be void of memory (it will represent a function), while AGI must have memory as we said above.

Partial Observability

We have cases where AI must necessarily have memory. Let us have some lived experience in the form of the following sequence of actions and observations:

$$o_0, \quad a_0, \quad o_1, \quad a_1, \quad \ldots \quad , o_{t-1}, \quad a_{t-1}, \quad o_t$$

Let the world change its internal state and let part of its internal state be hidden (not visible in the current observation). Suppose the hidden part is important (i.e. the future development depends on it). Let the hidden part depend from the past. Then AI must have the capability to memorize (perhaps not everything, but there will be important things, which are worth to memorize). Accordingly, we will have to assume that AI is a device with memory (or a function with memory). Therefore, this is a function which takes as input the current value of its memory p_i and the current observation, and returns the next action and the new value of its memory (we will call it *preliminary memory* because it does not include the last observation).

$$f(p_i, o_i) = \langle a_i, p_{i+1} \rangle$$

Now we will supplement the sequence of actions and observations by adding the values of AI's preliminary memory as well as the internal states of the world:

$$p_0, o_0, s_0, a_0, p_1, o_1, s_1, a_1, \quad \ldots \quad , o_{t-1}, s_{t-1}, a_{t-1}, p_t, o_t, s_t$$

The description of the world will be derived from the initial state s_0 and a function g which takes as input the current state of the world and the corresponding action, and returns the next observation and the new state of the world.

$$g(s_i, a_i) = \langle o_{i+1}, s_{i+1} \rangle$$

Note: We will note here that s_0 may depend on o_0. (This would be the case if the state "remembers" the last observation.) So we will say that the world is described by o_0, s_0 and g. In the next paragraphs we will introduce the notion of "intermediate state of the world" (n_i) and will split the function g in two functions, g_3 and g_4. Thus, the description of the world will be derived from the first intermediate state n_0 and the functions g_3 and g_4.

While in the *Full Observability* approach we sought to approximate the function *f*, we will now approximate *g*. In other words, instead of trying to describe the expert (whose lived experience we have), we will try to understand (describe) the world. In the case of *Partial Observability* we cannot approximate the expert because may not necessarily have the lived experience of an expert, but the lived experience of a random person. It may be our own lived experience which is stained with many errors. Such lived experience would not be an adequate role model.

If we are to understand the world, we must find the function g and the current state s_t. However, we cannot find these exactly so we have to resort to their approximations. Therefore we will find the function g' and the state s'_t which are the approximations of g and s_t.

The memory of AI will consist of g' and s'_t. (We assume that the function g is constant and does not depend on t, but function g' will depend on t because it presents our current understanding of the world and that understanding changes over time.) We will forget about the AI memory as such and will explore the sequence from which the AI memory is excluded:

$$o_0, s_0, a_0, \quad o_1, s_1, a_1, \quad \dots \quad , o_{t-1}, s_{t-1}, a_{t-1}, \quad o_t, s_t$$

If we manage to create a program which understands the world (is able to find g' and s'_t), we will tell this program to explore the future (take several steps forward) and choose the action which takes the world on the path to best development (which is the best-development scenario is yet to be discussed). This program would be exactly the AI we are looking for.

The Intermediate State

When it comes to transitions between the internal states of the world, we may reckon that after the action and the observation the world leaps straight in its next internal state. But, we may take another train of thought and assume that before accomplishing its next internal state the world goes through one intermediate state. We can imagine that the action and the observation are two distinct events which cause two subsequent changes of the state of the world. Let us assume that after the agent's action the world moves to a new state which reflects the result of that action. Let this be the *intermediate state*. Then the world changes itself one more time (and does it alone, i.e. without input from the agent) and thus reaches its next internal state. When the second change occurs the world generates the corresponding observation and transmits it to the agent. The term we will use for all internal states which are not intermediate ones is *steady states*.

Let us illustrate this with the world of chess where the steady states would be the positions on the chessboard when the agent is to make the next move. We can add intermediate states and these will be the positions when the world (the opponent) is to make the next move. Such intermediate state will reflect the agent's move but will not reflect the next move of the world (of the opponent who lives in the world).

We will assume that intermediate states and steady states combine to form a single set because the difference between them is not essential. In a chess game for example, if we look at a particular position on the chessboard in the majority of cases we will not be able to tell who made the last move and who is to make the next move.

Thus, in the sequence of actions and observation we can insert intermediate states between the steady states of the world:

$$n_0, o_0, s_0, a_0, n_1, o_1, s_1, a_1, \quad \dots \quad , o_{t-1}, s_{t-1}, a_{t-1}, n_t, o_t, s_t$$

Now we can split the function g in two functions: g_3 and g_4. (We already divided it in g_1 and g_2 but that was a different split.)

$$g_3(s_i, a_i) = n_{i+1}$$
$$g_4(n_{i+1}) = \langle o_{i+1}, s_{i+1} \rangle$$

Here g_3 only reflects the change of the world's state resulting from the corresponding action in the sequence. The result is an intermediate state which is then taken by g_4 to produce the new observation and the new steady state of the world. (The g_3 function "memorizes" the last action, but full memorization occurs only when g_3 is an injective function.) The function g can be expressed by g_3 and g_4 as follows:

$$g(s_i, a_i) = g_4(\, g_3(s_i, a_i)\,) = \langle o_{i+1}, s_{i+1} \rangle$$

As you can see, the intermediate states of the world correspond to the preliminary memory of the agent. We will introduce m_i – this is the steady memory of the agent which will reflect the last observation. Thus, m_i will memorize (not necessarily fully but only partially) the preliminary memory and the last observation. The steady memory of the agent will correspond to the steady state of the world.

$$p_0, o_0, m_0, a_0, p_1, o_1, m_1, a_1, \quad \ldots \quad , o_{t-1}, m_{t-1}, a_{t-1}, p_t, o_t, m_t$$

In order to play its next move, the agent will only need its steady memory. Certainly, the agent must also know how the world works (i.e. it must know the function g). For this reason we assume that our steady memory consists of our understanding of the steady state of the world and our understanding of the g function (the understanding we have at a particular moment of time).

The way to describe the current state

Let S be the set of internal states of the world (steady and intermediate). This is a set we know nothing about.

Let S' be the set of imaginary states of the world. This is one particular set chosen by us. We will use this set to approximate the set S. The set S' may change over time, but for any particular moment t we will have chosen a particular set, and at that moment this will be our idea of the possible states of the world.

The reachable states of the world are countably many (starting from s_0). We can therefore assume that the set S' is countable. Any encoding of S' is possible so we can assume that $S' = \mathbb{N}$. We can even assume that

$$s'_i = i$$

This is one possible solution, but it is not a good idea. First, such encoding depends on t and at the next step we will have to change it. The more serious problem is that in this way we provide an ultimately simplified representation of the internal state of the world and move all the complexity to the function g'. Any permutation (encoding) of the internal states of the world is possible, but the encoding process can lead to complication. Conversely, it would be better to have a more complex structure of the internal state and thereby keep the function g' more simple.

Let us take a program written in C++ or Java. What does its internal state look like at the moment of its execution, i.e. moment t? Such state is described by the current values of the variables. Thus, this internal state can be presented as a n-tuple of scalars. We can therefore assume that $S' = A^n$ (here A is the set of possible values of the scalar). Let us add that besides variables programs have arrays. Therefore, we will replace some scalars by k-tuples. This gives us a finite set S' which describes the internal state of a particular program (the values of its variables and arrays). This would be the case if the value of the variables is limited (e.g. 64 bits). If the arrays can change their size during the execution of the program or if the value of the variables is not limited, then S' will be countable (i.e. $n = \infty$ or $A = \mathbb{N}$).

Programs have countably many internal states because they use finite memory (even the Turing machine uses only a finite portion of its tape at each point of time). If we wish to describe a world that uses infinite memory we will have to assume that the set S' is uncountable. Imagine a Turing machine which starts running with a tape that is full of information all over (rather than a finite part of the tape, as is the usual case). In order to describe the internal state of such a machine we would have to imagine uncountably many internal states which correspond to the states of the infinite tape, and these are continuum many.

We will therefore abandon the assumption that the set of imaginary states S' is countable and will assume that S' describes the values of variables and arrays, wherein the arrays can be either finite or infinite.

How will then the state s'_t look like? It will not be a particular element of S'. Instead, s'_t will be a description of a set of states of S'. For example, if we have an infinite tape full of symbols, then the description of the state of that tape can be a specific value for the first 10 symbols and an unknown value for the remaining symbols.

Large Language Models (LLM)

Imagine we are having a conversation with someone and need to answer the next question. To do so, we must bear in mind the whole conversation, not just the last question. In this case our AI should not be a function which gives the answer to the last question only. However, even in this case we manage to reduce the problem to *Full Observability*. For this purpose we will simply assume that the observation is not limited to the last question, but encompasses the entire conversation up to that moment. The expert's lived experience looks like a set of 2-tuples wherein each tuple consists of the conversation up to that moment including the last question and an answer (some continuation of the conversation) $\langle conversation, continuation \rangle$.

Certainly, in this case if we search for AI as an approximating function, we have to deal with an overwhelming number of conversations and their continuations, and these continuations have to be the right continuations, i.e. they have to be provided by an expert who says the right thing. Given the above assumption that we are going to replicate an expert, we must make sure that the lived experience at hand is the one of somebody who is an expert indeed.

Some argue that LLM is not a function because rather than returning just one possible continuation for each start of a conversation, it returns countless possible continuations. The truth is that it is a function which returns a probability distribution (a table which maps the various continuations to their respective probabilities). Then we choose randomly a certain continuation (based on the distribution). To put it more simply, imagine a function which returns a table with k

rows such that for row i you have probability p_i. Then you flip a coin to determine j (with probability p_j). Now, a function combined with randomness is no longer a function, but the essential part is a function.

An LLM example

Let us exemplify an LLM by creating a program that plays chess without even understanding what the position of the pieces on the board is. In order to describe the world, we need to identify the agent who lives inside this world and moves the pieces against us. Let our opponent be a program which calculates three moves ahead and chooses the two best moves. Then the program flips a coin to decide which of these two moves to play. (We can of course assume that the opponent does not flip a coin and always plays its best move, but this would make the world deterministic, which is too elementary. In a deterministic world all we need is to win once since this will enable us to repeat the winning game infinitely.)

Let us present each chess move as a 4-tuple $\langle x_1, y_1, x_2, y_2 \rangle$ where $\langle x_1, y_1 \rangle$ are the coordinates of the square from which we lift the chess piece and $\langle x_2, y_2 \rangle$ are the coordinates of the square in which we put the lifted piece. There are 8 possibilities for each coordinate. Therefore, all possible moves are $2^{12} = 4096$, but only a small fraction of all possible moves are correct moves.

Let *action* and *observation* be the moves of the expert and of its opponent (the same 4-tuples).

Let the expert whose lived experience we are going to replicate as closely as we can be the program which calculates ten moves ahead and chooses the best move. Let the lived experience consist of a large number of games between the expert and the opponent (each game, together with all of its beginnings that end with expert's move). In the 2-tuples, the action will be the last move of the expert and the observation will be the game as it has unfolded from the beginning until that move.

The approximation in this case will not be of the function f, but of the function f^*:

$$f^*(\langle o_0, a_0, \ldots, o_{i-1}, a_{i-1}, o_i \rangle) = a_i$$

The function f^* is natively definable by f and p_0. The function has only one argument, although this argument is a list of many moves.

If we wish our function f^* to play reasonably well, we should have a huge amount of lived experience (a huge number of games played between the expert and the opponent). It would be very difficult to even create a program that plays correct moves only. In order to play correct moves the program must be able to understand the position on the chess board which is an enormously challenging task.

This is a problem with the *Full Observability* approach. Learning in this approach requires a huge amount of lived experience. This means that the program is a very slow learner (slow not in the sense of time, but in terms of the amount of information or the number of steps contained in the lived experience). This is not the case with human beings. This might be true at a lower level where humans perceive visual and auditory information, but at a higher level things are different. It only takes a few examples for a person to figure out some pattern. It is often sufficient to tell somebody something just one time and he will already know it forever and will be able to use it.

If we can find a function f^* which plays only correct moves, it will probably play quite strong moves because it will replicate the expert, and the expert is a good player. That is, the key is to understand the state of the world (the position on the chessboard). Once we achieve this task, we will get a strong player as a side effect.

Dividing life in games

Very important in the above example is that we divided the life (lived experience) in separate games which start from the same internal state of the world (the starting position on the board). With this division, the order in which the chess games are played does not matter. In this way we can represent lived experience as a tree with many branches.

If lived experience is a sequence of observations and actions that cannot be divided in separate games, then lived experience would be presented as a straight path without any branches (or a tree without branches).

Dividing life in separate games is very important if we are to find f^* by approximation. The argument of this function is the current game, and if we have only one game, then the argument will comprise all lived experience which means an overwhelming amount of information. Approximation will be made easier if lived experience is an abundantly branched tree because in that case we would have more information about how the world and the expert behave.

Can we divide lived experience in separate games? To do this, we need to find moments when the world is in the same state. Basically, in Event-Driven (ED) models [1] we make a classification of states and divide them in groups of similar states. It goes without saying that defining two states as similar does not in any way suggest that these two states are identical. We can perfectly assume that all states in life are different. As Heraclitus put it, "You can't step in the same river twice".

One can also assume that life consists of separate games, but they do not begin from the same state. Lived experience then looks like separate unconnected threads. This assumption would make understanding the world even more difficult because before starting a new thread we have to find our whereabouts and figure out what the current state of the world is.

I was told the story of a drunkard who lived two parallel lives. In one of his lives he was sober all the time and in the other life he was drunk all the time. He only remembered what happened while he was sober and so his life was made of separate unconnected threads. Each time he became "aware" (when he started a new thread) he had to find out what the current state of the world is (where he was, what his health and financial situations were and so on). Of course, the current state of the world changes in a leapfrog manner, but the function g remains constant (e.g. the Earth gravity remains unchanged). It is not quite clear what part of the world is described in s_t and what part is reflected in the function g. For example, if the drunkard became "aware" in outer space then even the Earth gravity may not be there.

Anton's question

My colleague Anton Zinoviev [9] asked me the following question: "Instead of approximating the function f^*, can we use approximation to find the function which describes the world, namely function g^*?"

$$g^*(\langle o_0, a_0, \ldots, o_{i-1}, a_{i-1}\rangle) = o_i$$

The function g^* is natively definable by g and by n_0. Certainly, g^* will be a multivalued function because the opponent chooses one of the two best moves. This is not an issue because we generally assume that the continuation in LLM is a multivalued function.

If we have g^*, we can use it to find the function f^* (by taking a few steps into the future). In other words, the question is whether we can use approximation in order to find an explanation of the world. The answer depends on the kind of lived experience we can rely on. Let us consider two cases:

1. In the first case, we have the lived experience of an expert. In this case, why approximate g^* in an effort to arrive indirectly to f^*? It would make more sense to approximate f^* and find it straight away.

We should note that in this case it would be difficult to find g^* by using the lived experience of the expert since the expert's lived experience is very scarcely branched (in each position we would have only one possible move of the expert). This information may not be sufficient for us to describe g^*, although we can use analogy to guess what will happen if we play a move other than the expert's move (the approximation will provide us with such an analogy).

For example, if the expert avoids making a particular move we cannot tell whether he avoids it because the move is stupid (bad for expert) or because it is incorrect (forbidden). In a chess game we are forbidden to make a move if our king will be captured in the next move. Although simple (single-move) errors in chess are prohibited, we can still play a move which will be followed by inevitable checkmate (our king will be captured in two moves). The expert would not play such a move and so we can decide that errors which encompass two moves are also forbidden. It does not matter a lot whether stupid moves are permitted or forbidden. In this way the world may be not well defined but the difference would not be essential.

A more essential difference would arise if the expert avoids a certain move not because it is a stupid move but because for some reason the expert dislikes that move. For example, imagine an expert who refuses to travel abroad. Then, if we rely solely on the lived experience of that expert we will describe a world where the only country that exists is the expert's home country, and all other countries merely do not exist. (Let us assume that the expert is sufficiently competent and in his home country he does very well.) Therefore, if we are guided solely by the expert's lived experience we may fail to describe a major part of the world.

The limitation would be even stronger if the expert is deterministic. A nondeterministic expert would gain richer lived experience, however, there will still be moves which he avoids. If we had the lived experience of a deterministic experts then there would be one strange but possible explanation of the world in which the agent's actions do not matter at all. In this situation we can assert that even if the agent plays some other move, the outcome would be the same.

Let us take the world of the chess game described above. Imagine a second world in which the intermediate state is the same regardless of the agent's move. Namely, it will be the state that would occur in the first world if the expert had chosen to make the move in question. Let in both worlds the opponent be the same (the same function g_4). The lived experience available to us (the one of the expert) would be the same in both worlds. That is, lived experience in this case would

not differentiate these two worlds. Nevertheless, the second world is more complex than the first one because it includes a description of the expert while the first world would not depend on the expert. Therefore, the second world is less probable than the first one (Occam's razor).

2. In the second case, let us have the lived experience of the random player. We can also assume that we have the lived experience of untrained AI which walks astray while exploring the world and makes many mistakes in the learning process.

In this case, the input should include four additional observations: *win*, *loss*, *draw* and *incorrect move*. We will assign scores (rewards) to these observations. *Win* will attract a positive score, while *loss* will bring about a negative score. The score of *incorrect move* will be the lowest because we should prefer losing the game instead of playing an incorrect move. The score of *draw* may be positive or negative depending on whether we aim for a draw or try to avoid a draw. The score of the remaining observations (all correct moves of the opponent except those which put an end to the game) will be zero.

We did not need these additional observations when we were trying to replicate the expert. The expert and the world could not play incorrect moves, and when the game ended we did not need to know how it ended because we were just following the expert and he knew what to do. However, now the g^* function will have to predict these observations so that we can choose the best course of actions.

The game can also end in a move made by the opponent. In this case we can tell AI that the only correct move is to concede loss (or draw). To avoid adding more actions, we will assume that in this case all AI moves are correct and the next observation will be *loss* (or respectively *draw*).

This is how the four additional observations will provide us with lived experience by which the function g^* can be approximated. In this case we will need huge lived experience, much bigger than the one needed in the fist case.

Compression

Generally we can assume that the state of the world is the current game as unfolded until present (or the entire lived experience if we cannot divide life in separate games).

$$g(\langle o_0, a_0, \ldots, o_i \rangle, a_i) = \langle o_{i+1}, \langle o_0, a_0, \ldots, o_i, a_i, o_{i+1} \rangle \rangle$$

While the current game carries all the information we need to determine the state of the world, that information is too extensive and unprocessed. The information needs to be compressed and strongly truncated. That compression should include loss of information because we need to extract everything essential and discard everything unimportant (we are not able remember all the information).

What is essential and what is unimportant? That will be determined by the model of the world we are going to find. The essential part will be the internal state of the world and the unimportant part will be how we have reached that internal state.

The g function

As we said before we may assume that the set of imaginary states S' is countable. The sets of actions (Σ) and observations (Ω) are finite (or countable). Therefore g' – which approximates g – is a function between countable sets.

$$g' : S' \times \Sigma \rightarrow S' \times \Omega$$

With some encoding we can think that g' is a function which takes as input natural number and returns natural number.

$$g' : \mathbb{N} \rightarrow \mathbb{N}$$

The problem is that the number of these functions is continuum many. We will be seeking a description of function g', and that description must be sought in some countable set of describable functions. The most suitable candidate for such a set is the set of computable functions. This approach is taken in [7, 6, 2] where the assumption is that the function of the world is approximated by a computable function.

Nevertheless, looking for g' in the set of computable functions is not a good idea because it implies that the world is deterministic. It is much better to assume that there is randomness in the world and that the function g' can call an oracle who would give a random value with some distribution (e.g. by rolling a dice). Adding such an oracle takes us away from the set of computable functions, but gives us a countable (describable) set that is convenient for predicting the future. Certainly, such a prediction would have variants and different variants would come with different probabilities.

There is another reason why we have to break away from the class of computable functions. There may be many agents operating in the world and the function g' may thus depend on the actions of other agents. We will present these agents as oracles which describe their behavior. These oracles are some unknown functions and adding them greatly simplifies the description of the world. Imagine how simpler it would be if the opponent in the chess example was just a black box (an oracle who seeks to play against us). Instead, we have described a particular program with randomness. Furthermore, if we a have a particular world with a particular opponent, it is very unlikely that we would manage to describe it as a particular program with randomness. Therefore, it is much easier and more reasonable to present the opponent as a black box.

The class of programs with agents is countable (describable) and has the nice property that we can use it to predict the future. Again, we have different variants for the agents' actions, but we can assume that some agents are willing to help us (friendly agents) while other agents seek to disrupt us (hostile agents). Hence, predicting the future will look like the Min-Max algorithm, where we will compute the maximum over the possible actions of the friendly agents and the minimum over the actions of the hostile agents.

Event-Driven models

As we said we are going to look for g' in the set of computable functions with oracles (functions which return randomness or emulate agents). Hence, we will represent g' as a Turing machine or as a program in some programming language (we will regard these oracles as external programs).

Every programmer knows that programs are quite fragile and changing one line or even one character will ruin the program. A program is like a complex clockwork that has many cogwheels, so changing the size of just one cogwheel will block the clock.

We would like to construct g' by using components that can be easily combined and changed. The suitable components from which we can construct g' are called Event-Driven (ED) models and are described in [1, 3]. ED models are akin to finite-state automata. For example, the head in a Turing machine is an ED model.

Based on ED models, we will construct the imaginary state of the world. For each ED model there will be one corresponding scalar, namely the current state of the ED model. In addition, for each ED model there will be one corresponding array (which can be finite or infinite depending on the number of states in the ED model). The state of the model will index the array and the values will be observed when the current state of the ED model is the one that indexes the corresponding value. The values of these scalars and of these arrays will describe the state of the world.

Algorithms

The algorithms will also be ED models. This is described in [5] where algorithms are used to define the rules by which the chess pieces move.

In [8] Yann LeCun says that nobody knows how to make a hierarchical algorithm. Actually, the problem is to make an algorithm, whereupon it would not be difficult to organize a hierarchy among algorithms. The making of an algorithm has already been described in [5].

Algorithms are important for planning the future. We said we will mentally take a few steps forward to choose the best development for the world. The point is that we will make large steps rather than small ones. A small step consists of one action and the transition from t to $t+1$. A large step is the execution of an algorithm which usually takes many small steps.

The reason why we prefer large steps is that this enables us look much further into the future. For this planning we will need a generalization of g'. The generalized function g' will take as input a certain idea of the state of the world and an algorithm, and will return an idea of the state of the world after the algorithm is executed.

The algorithms can also be executed in parallel. In our daily life we typically run multiple different algorithms in parallel. For example, we may read a newspaper while waiting for a bus. Meanwhile we may also be doing our studies in the university (this is also an algorithm consisting of attending lectures and taking exams).

While waiting for the bus we can also plan our day. Planning is also an algorithm. Most things in AI happen in a decentralized or parallel manner. For example, search for patterns, planning and execution of algorithms can occur unconsciously or at a subconscious level. There is however one thing that has to be centralized and conscious – namely choosing one of the plans made or executing one of the algorithms. (This is the case when we have a choice. If there is no choice, we do not need to bother about it.) For example, two plans pop up in our mind. One is to go to the university and the other is to go for a walk in the park. We have to consciously (centrally) choose one of these two plans and start executing the corresponding algorithm. Otherwise, very much

like Buridan's donkey we will end up in a deadlock between the university and the park and we will not be able to go anywhere.

Conclusion

In order to describe the state s'_t and the function g' we need a language for description of worlds. Such a language has been described in [4, 5]. The language is based on ED models and abstractions such as agents and objects.

Can we create AGI by using LLM? In [8] LeCun says that this cannot be done because every LLM computes a finite function. (More precisely, he says that the time to find the answer to a question is constant and does not depend on the question, but this is the same thing put another way.)

We completely agree with LeCun that we cannot create AGI through LLM, but differ in our reasoning. The obstacle is not that LLM is a finite function, but because it is a memoryless function and because we learn by trying to replicate some expert rather than learn from our own lived experience.

The fact that we have agreed to construct g' from ED models means that we have abandoned the search in the set of computable functions and will be happy to go only with finite functions. Actually, looking for g' among the computable functions is not a very good idea. In this case we would get an AI which is prone to seizures (it will get stuck and remain deadlocked for a while). We will therefore assume that the time it takes to compute g' is limited by some maximum. Calculating g' means to take a particular idea of the state of the world and an action of the agent, and obtain the new observation and the new idea of the state of the world (this happens at given values of the oracles).

Although function g' is finite, it can compute anything (any program). All we have to do is give AI an infinite tape and the ability to read and write on it. In this way we will embed an arbitrary Turing machine in g'. Similarly, the head of Turing's machine can execute any program, but does this in many steps rather in a single step. The idea of giving AI an infinite tape matches Turing's idea that his machine is a human being provided with infinite amount of paper.

Therefore, for a single step we have a finite function, but for multiple steps we get a computable function. Nevertheless, we wish g' to be something more than a finite function and that g' does not get stuck. Accordingly, we will assume that we have oracles and these oracles compute programs. We already added oracles that roll dices and emulate agents. Nothing prevents us from adding ones that compute programs, but to ensure that g' does not get stuck we will assume that these oracles operate asynchronously. That is, if they manage to return an answer at the same step, that will be fine, but if they do not, they may return an answer at step $t + 1$ or at some later step, and until then the answer will be *unknown*.

We mathematicians know that proving a theorem may take years, and sometimes there are theorems we cannot prove it at all. Proving theorems is more akin to multi-step execution of algorithms, while the asynchronous execution of programs resembles a situation where we cannot answer a question right away, but in a while the answer pops up subconsciously in our mind.

We said in the beginning that the goal of AI is to predict the future. However, this is only its first goal. We need to add a second goal, namely to determine which future is better and which is worse. We will call this second goal *the meaning of life*. The behavior of AI is determined by this second goal and by something else which we call *character*. The meaning of life we will impart in AI matters a lot. Equally important is what character AI will have, or to put it simply, what kind of guy AI will be. We will address these questions in our next paper the title of which will be *How Can We Make AI with a Nice Character*.

References

[1] Dobrev, D. (2018). Event-Driven Models. *International Journal "Information Models and Analyses", Volume 8, Number 1, 2019, pp. 23-58.*

[2] Dobrev, D. (2019). The IQ of Artificial Intelligence. *Serdica Journal of Computing, Vol. 13, Number 1-2, 2019, pp.41-70.*
https://serdica-comp.math.bas.bg/index.php/serdicajcomputing/article/view/sjc.2019.13.41-70/pdf

[3] Dobrev, D. (2021). Before We Can Find a Model, We Must Forget about Perfection. *Serdica Journal of Computing, Vol. 15, Number 2, 2021, pp. 85-128.*

[4] Dobrev, D. (2022). Language for Description of Worlds. Part 1: Theoretical Foundation. *Serdica Journal of Computing 16(2), 2022, pp. 101-150.*
https://serdica-comp.math.bas.bg/index.php/serdicajcomputing/article/view/sjc.2022.16.101-150/pdf

[5] Dobrev, D. (2023). Language for Description of Worlds. Part 2: The Sample World. *Serdica Journal of Computing 17(1), 2023, pp. 17-54.*
https://serdica-comp.math.bas.bg/index.php/serdicajcomputing/article/view/sjc.2023.17.17-54/pdf

[6] Hernández-Orallo, J., & Minaya-Collado, N. (1998). A formal definition of intelligence based on an intensional variant of Kolmogorov complexity. *Proc. intl symposium of engineering of intelligent systems (EIS'98), Feb.*
1998, La Laguna, Spain (pp. 146-163). : ICSC Press.

[7] Hutter, M. (2000). A Theory of Universal Artificial Intelligence based on Algorithmic Complexity. *arXiv:en.AI/0004001 [en.AI]* https://arxiv.org/abs/cs/0004001

[8] LeCun, Yann (2024). Lex Fridman Podcast #416. https://youtu.be/5t1vTLU7s40

[9] Zinoviev, Anton. https://www.fmi.uni-sofia.bg/en/faculty/anton-kirilov-zinoviev