**The difference between AI and General AI**

General AI

Artificial General Intelligence has reached parity with human intelligence. An AGI system can perform any task that a human is capable of performing. AGI systems can also perform tasks more efficiently than humans, only for a specific function assigned to them. But, they have no capability to perform any task that is not assigned to them. A human is less skilled at the task but can also perform a wider range of functions than AI applications.

Artificial intelligence (AGI) is a type of AI that we see in movies. AGI is also a machine that can solve any problem by displaying the general intelligence of a human. Strong AI and deep AI are two terms for artificial general intelligence. This is a concept of a machine with general intelligence that can think, understand, learn. It can also apply its intelligence to solve any problem in any situation like humans.Strong AI employs the theory of mind AI framework. It also helps to train machines to recognize needs, emotions, beliefs, and thought processes in humans, rather than replicating or simulating them. Researchers working on AGI must figure out how to make machines conscious. This has to be done by programming them with a full set of cognitive abilities.

Narrow AI

Narrow artificial intelligence (narrow AI) is a type of artificial intelligence. In this, a learning algorithm performs a single task. Any knowledge gained from that task is not automatically applied to other tasks. Narrow AI completes a single task without the need for human intervention. Also, language translation and image recognition are two popular narrow AI applications. The vast majority of AI applications in use today begin as narrow AI. Narrow AI, also known as weak AI, is a type of artificial intelligence with limited scope. I hear the term Turing Completeness thrown around a lot more in software engineering than I would expect. I think there’s a little confusion about what Turing Complete means, especially within a software engineering context. We say that Turing Completeness is a property of a computational system that states that the system is as computationally powerful as a Turing machine. But what EXACTLY does that mean? Put your scuba gear and wetsuit on because we’re diving deep into formalism:

**Turing Completeness**

Turing Machines

Turing machines are theoretical computers defined by Alan Turing in his highly influential paper titled On Computable Numbers, with an application to the Entscheidungsproblem, published 1936. Turing machines are abstract mathematical constructs that help us describe in a rigorous fashion what we mean by computation.

There are countless in-depth tutorials on how a Turing machine works, and you can watch any of them! I can pretty much guarantee that they’re going to be better at explaining how a Turing machine works than me, but in case you don’t want to watch any of those, I’ll throw my description in as well. A Turing machine consists of 2 elements: The computational head and an infinitely long tape. The head operates roughly as a ‘read-write’ head on a disk drive, and the tape is divided up into an infinitely long set of squares, for which on each square a symbol can be written or erased. The Turing machine recognizes and can write down a finite set of symbols, called the Turing machine’s alphabet.

The Turing machine is only ‘aware’ of one square on the tape at a time — namely the square the head of the Turing machine is currently on.

On that tape, a Turing Machine can do any of these 4 actions:

\* Move the head left by 1 space

\* Move the head right by 1 space

\* Write a symbol at the head

\* Erase a symbol at the head

The machine decides which of these operations to do on any given step through a finite state machine. Different Turing machines have different state machines that define their operation, that means Alan Turing had to create different machines for different programs. So he created "Universal Turing Machine" that can take ANY program and run it.Programming languages are similar to those machines (although virtual). They take programs and run them. Now, a programing language is called "Turing complete", if it can run any program (irrespective of the language) that a Turing machine can run given enough time and memory.

For example: Let's say there is a program that takes 10 numbers and adds them. A Turing machine can easily run this program. But now imagine that for some reason your programming language can't perform the same addition. This would make it "Turing incomplete" (so to speak). On the other hand, if it can run any program that the universal Turing machine can run, then it's Turing complete.

**Discuss whether Bitcoin is Turing Complete**

Bitcoin is not Turing complete .Satoshi chose Bitcoin complexity keeping in mind the intended applications functionality. Making Bitcoin Turing complete would have meant having to provide for looping statements and hence even possibly infinite loops. Non Turing complete systems and networks are designed to always be in a deterministic state. Developers and ecosystem users do not have to about infinite loops causing resource drains and crashing systems.

An optimal program should be written at a minimum level of complexity to maximize performance and minimize issues. Turing completeness is theoretical, nothing is Turing complete in practice. You can think of contracts as predicates. Essentially it evaluates to true or false — which indicates if the coins can be moved or not. With bitcoin, we want strong guarantees that programs terminate quickly. This keeps nodes from being DoSed by malicious users. Remember, nodes are not compensated for computational power on the bitcoin network. Therefore we need to minimize CPU/memory usage for nodes.

**Discuss what role blockchain might play working with AI in the future**

Blockchain technology and AI are often cited together towards a common future. Blockchain provides a strong foundation for secure data storage and access on a mass scale. An AI that intends to be versatile and open-source, blockchains may be the best solution for handling data security and storage. Collecting and processing the said data is a big challenge too, both logistically and legally. This is the primary reason behind the open-source AIs being very crude, requiring rigorous training for any use case.

“Decentralization of public usage data has the potential to face resistance from internet privacy and security advocates. Private channels are able to achieve it through their products and willingly accepted terms and conditions. There is another case to be made on the kind of data that should be collected, which is often determined by the need to train the AI than the other way round. So yeah, any public AI project will have to contend with these limitations. The use of blockchains to store data is the only strong point that can work in its favor.” added Mr. Sharma of the Canada-based agency.

AIs and blockchain also have a big difference in their workings. Blockchains store data in a tamper-proof manner while AIs process and store patterns in the data. These can be extremely complimentary functions if used right or could break the system completely. Private institutions may not find advantages in integrating blockchains to AI systems. So currently, AIs and blockchains are following their own paths.

Industrial applications of AI for automation and operations can still use blockchains to their advantage like factories and supply chains employing AIs. Blockchains are already in use for many such industrial applications. Game AIs can also benefit from such a system especially when Metaverse and crypto games are on the rise already. Blockchain-based AIs can also find applications in public systems like traffic management, billing and taxing, project management, etc. Future of AI and blockchain is intertwined for sure but what will come out of it is still to be seen!