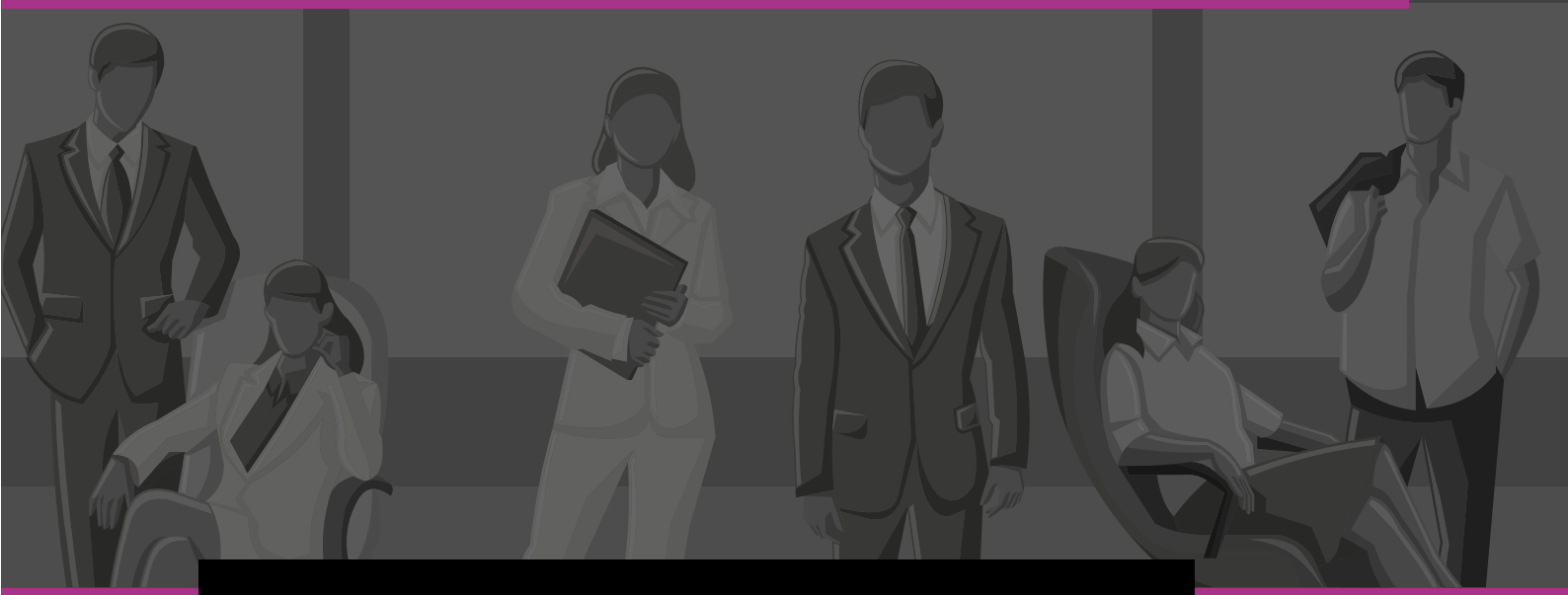
LOGIC.TOOLS

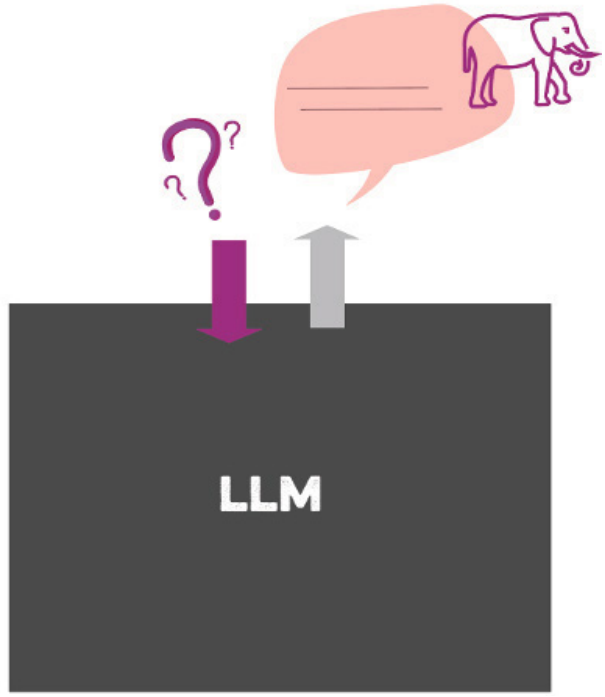


The Power of Why in your favourite LLM

Explainable | Verifiable | Accurate

The intrinsic limitations of LLMs

LLMs do wonders. They make sense of heaps of unstructured data and extract wisdom by finding deeply buried connections. Beyond the apparent magic, LLMs sometimes fail, produce erroneous answers and hallucinate non-existing inferences. No amount of testing can protect against this, as LLMs are intrinsically non-deterministic. **They are not designed for absolutes.** They come with the possibility of errors, at a rate which makes them unsuitable for many tasks.



This is crippling: many LLM-based applications are so uncertain that they end up requiring an expert to validate their output, which defeats the purpose altogether.

LLMs' magic also comes at a steep cost. Finding these connections in unstructured data requires energy as well as processing power and seriously increases response times. It works fine on a developer's desktop but becomes unsustainable when deployed at scale for general use.

Ironically enough, the most common solution to this consumption issue is to reduce the LLM's domain, in breadth as well as in depth, which in turn increases the chances of an erroneous answer.

This raises the question: how can one remedy this? **How can one build LLM-based systems that are both reliable and sustainable?**

MCP: divide and conquer

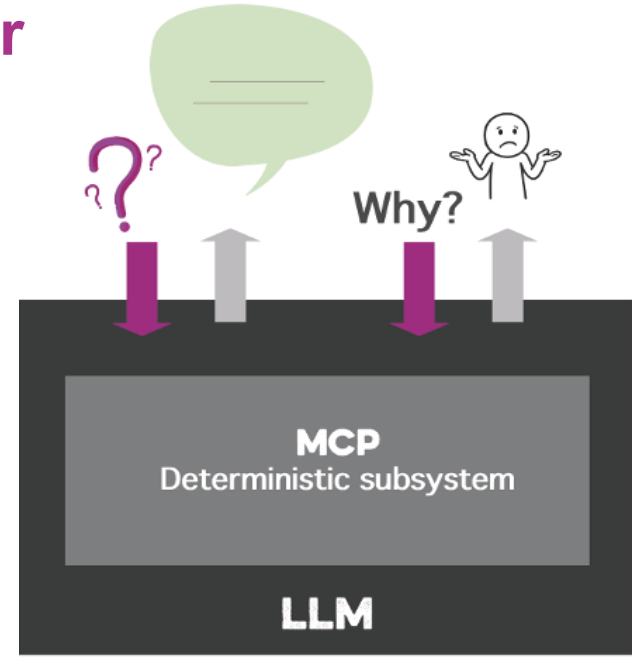
The Model Context Protocol is a mechanism by which one can connect LLM's to existing software components.

In essence, an MCP server publishes tools, together with a description, including the parameters they take. A properly configured agent can then call these tools based on their description at the request of an LLM.

It is generally understood that these tools should not be probabilistic. They should provide dependable and deterministic foundations on which LLMs can rely.

MCP servers operate as black boxes. They don't include the capability of justifying any result they provide.

This is their biggest limitation.



Logic.Tools for MCP

Logic.Tools can take over the bulk of the burden of the complex tasks that make LLMs stumble, by fully supporting the deployment of logic-based applications as MCP servers.
Such deployments cover the full breadth of the application, and reuse the existing artefacts extensively:

+ The **data model**, implemented as the domain **ontology,** is available for data-related queries.

+ The **rules are exposed** as individual tools, and the descriptions found in the literate definition of the system are used to describe them.

This setup provides the best of both worlds:

+ The **reasoning** provided by Logic. Tools is robust, **deterministic** and **explainable**. It does not hallucinate. It does not go awry unexpectedly. It is also efficient in resource consumption as well as in response times.

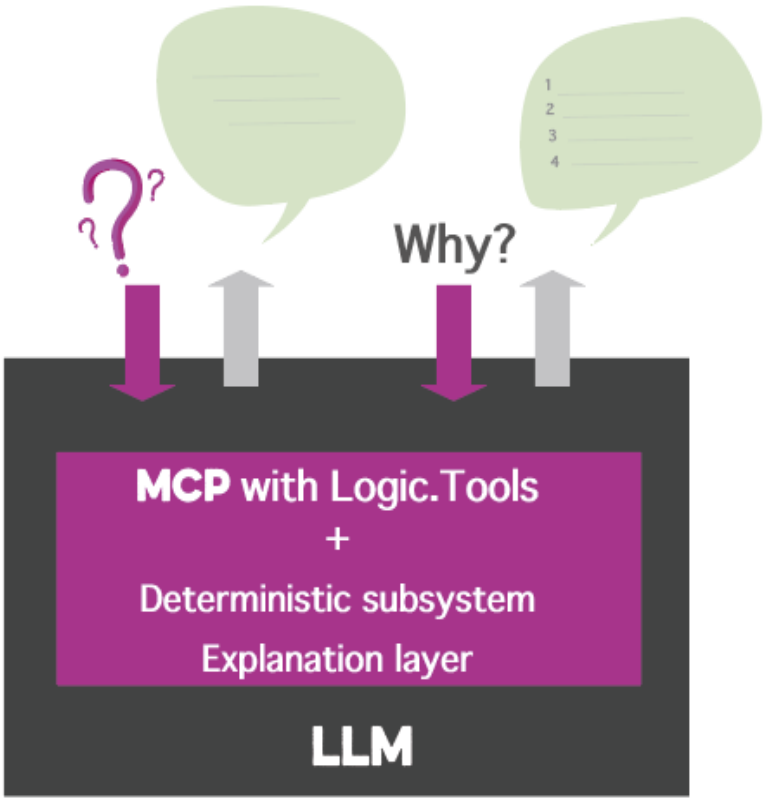
+ **LLM** do what they do best, namely **process natural language inputs** with little need for reasoning-like processing.

By dramatically reducing the scope for possible hallucinations, this scaffolding proves to be very effective for the deployment of production-level AI-based systems.

The power of why?

Unlike LLMs, Logic.Tools is fully deterministic, but so are most conventional programming languages, that can all be used to implement equally deterministic MCP servers.
So, what makes Logic.Tools special here?
There is of course **reasoning based on pure logic**, as opposed to all variations on imperative programming. Logic.Tools' reasoner **provides** exactly what LLMs don't, namely **determinism, traceability and dependability.**
There is something even more unique to Logic. Tools' approach to MCP servers, something missing in the most advanced RAGs, namely the ability to answer
the most obvious questions of all: "Why?"

"Why did this tool return this answer?"
This ability to **explain any result, in natural language, and at scale** (for all MCP tools published in this way) is unique to Logic Tools.



It leverages its existing explanatory layer for the chain of inference, with references to the rules that have been applied (instantiated with values), turned into a natural language description using an AI engine.
This allows the Logic.Tools system to be more tightly integrated in the LLM. Instead of behaving like a black box, it can justify and explain everything it does, every single claim it makes.

LOGIC.TOOLS
Semantic Intelligence

Your organization is defined by its software systems, and you cannot afford to have the essence of your business buried under myriads of technical details lost in huge amount of code.

Logic Tools provides you with the clarity and flexibility you need, and helps you master the complexity and the fast pace of evolution while keeping control at all times.

Semantic Intelligence

Address
Logic.Tools
Rue de la Caserne 45
1000 Brussels
Belgium

Phone
+32 2 522 06 63