Using Large Language Models and s(CASP) to build trustworthy AI systems

Gopal Gupta

UT Dallas Center for Applied AI and Machine Learning & ARCOS CLARISSA Team

Intelligence has two components: learning *and* reasoning. Machine learning & LLMs alone are not enough to achieve human-like AI. Automated reasoning is indispensable for achieving it. We have developed s(CASP), an automated commonsense reasoning engine, that can model human thinking. Coupled with machine learning & LLMs, it has been used to develop trustworthy & explainable AI applications.

What is intelligence?

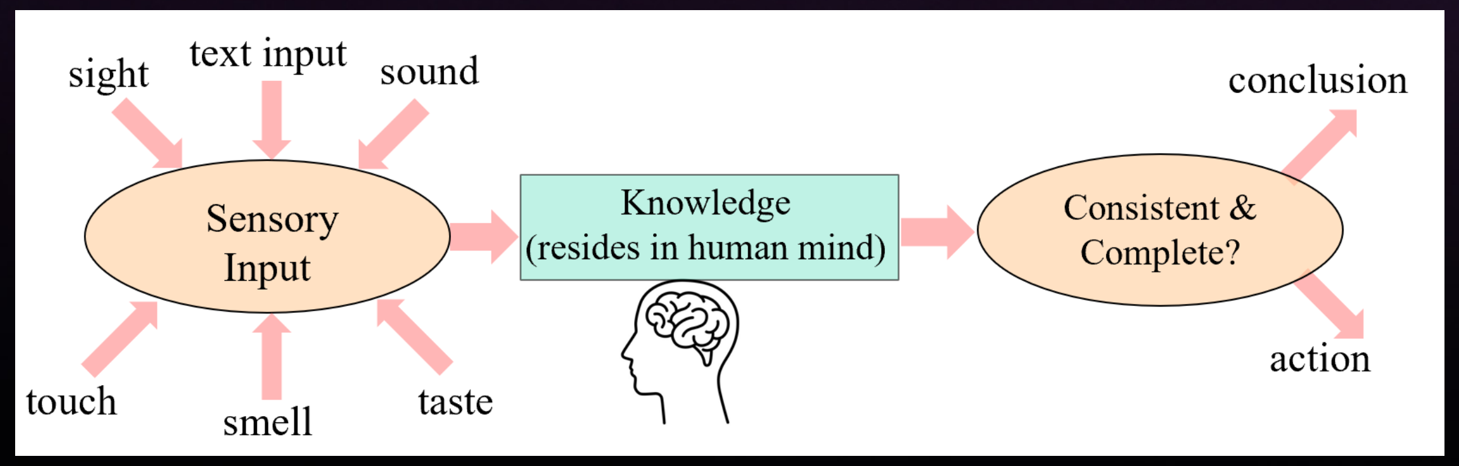
- Intelligence has two broad components:
  - Acquiring or learning knowledge (automate it: **machine learning**)
  - Applying knowledge that is learned (automate it: **automated reasoning**)
  - + Vision, Speech Processing, Speech Generation, etc., etc.
- Humans operate by using their senses (sight, sound, smell, taste, touch) and then by performing reasoning over the sensory input received
  - The information collected by sensing resides as knowledge in the human mind
  - This knowledge is then checked for consistency and completeness, and then a conclusion is drawn that spurs an action by the human

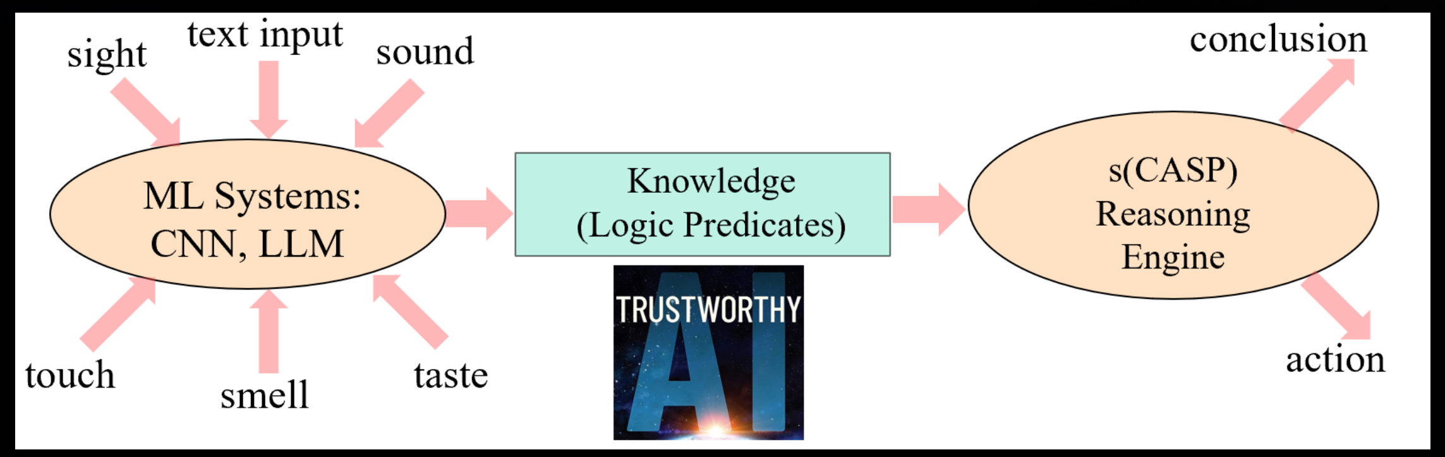
Sensing = Kahneman's System 1 or Fast Thinking;

Reasoning = Kahneman's System 2 or Slow Thinking

(System 1 & System 2 interplay in complex ways)

Today: Sensing/machine learning in limelight; reasoning not so much





Commonsense Reasoning

- We employ commonsense reasoning to explain things:
  - In deliberative thinking, we convert sensory input to knowledge, over which we reason
- To automate explainability/interpretability, ***automate deliberative thinking***
- Commonsense reasoning can be approximated with ***answer set programming***
  - default rules, integrity constraints, and assumption-based reasoning
- **Default Rules:** express what is true in a majority of cases but not always
  - e.g., "birds normally fly" OR "by default birds fly, but there are exceptional ones that do not"

```
flies(X) :- bird(X), not ab0(X).          ab0(X) :- penguin(X).
```

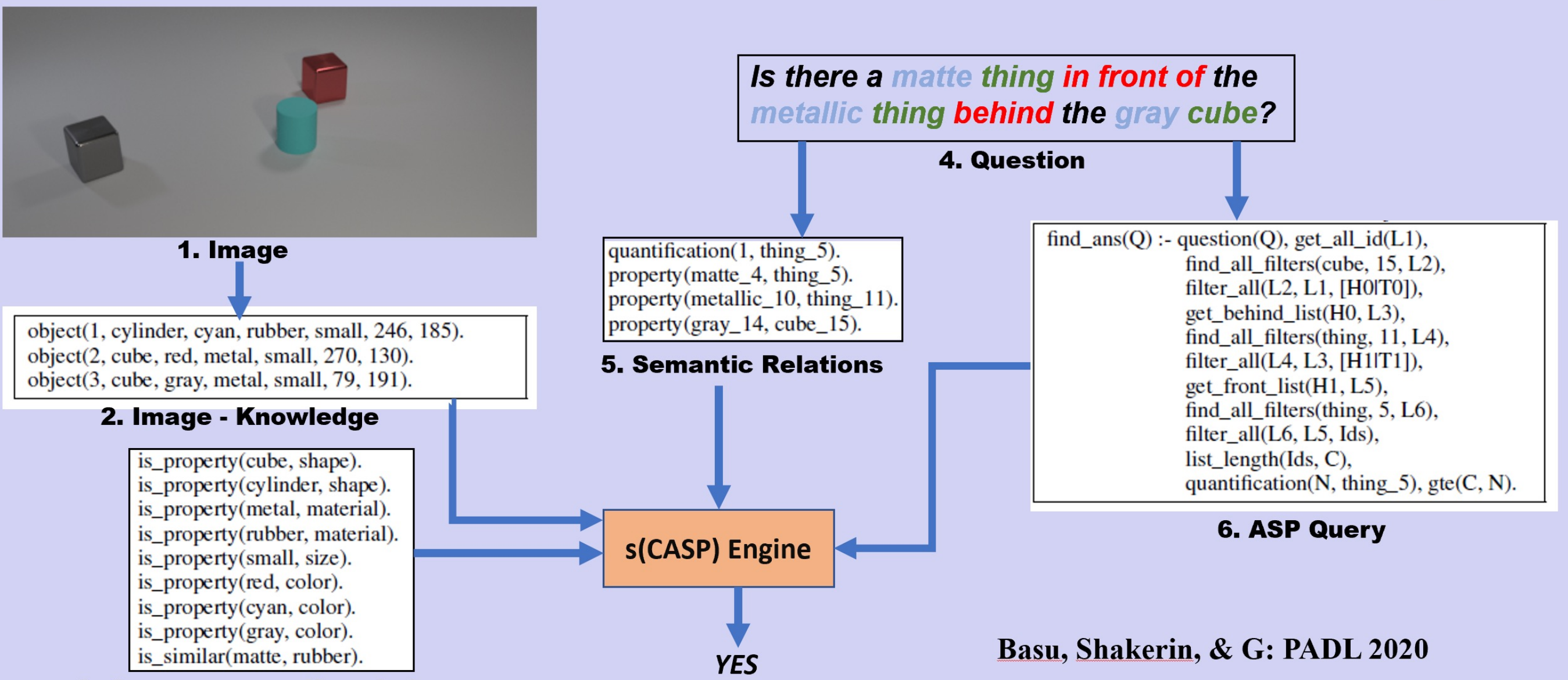
- **Integrity Constraints:** express impossibility conditions
  - A person cannot sit and stand at the same time; a living human must always breathe

```
false :- sit(X), stand(X).          false :- alive_human(X), not breathe(X).
```

- **Assumption-based reasoning:** if we don't know something, we will assume it holds (or does not hold) and continue (abductive reasoning)
  - Game of clue: for each suspect, assume they are the killer (or not) and reason through them

s(CASP) is a query-driven ASP engine that supports commonsense reasoning

AQuA: YOLO & ASP-based Question Answering



1. Image

2. Image - Knowledge

```
object(1, cylinder, cyan, rubber, small, 246, 185).
object(2, cube, red, metal, small, 270, 130).
object(3, cube, gray, metal, small, 79, 191).
```

3. Commonsense Knowledge

```
is_property(cube, shape).
is_property(cylinder, shape).
is_property(metal, material).
is_property(rubber, material).
is_property(small, size).
is_property(red, color).
is_property(cyan, color).
is_property(gray, color).
is_similar(matte, rubber).
```

4. Question

Is there a matte thing in front of the metallic thing behind the gray cube?

5. Semantic Relations

```
quantification(1, thing_5).
property(matte_4, thing_5).
property(metallic_10, thing_11).
property(gray_14, cube_15).
```

6. ASP Query

```
find_ans(Q) :- question(Q), get_all_id(L1),
               find_all_filters(cube, 15, L2),
               filter_all(L2, L1, [H0|T0]),
               get_behind_list(H0, L3),
               find_all_filters(thing, 11, L4),
               filter_all(L4, L3, [H1|T1]),
               get_front_list(H1, L5),
               find_all_filters(thing, 5, L6),
               filter_all(L6, L5, Ids),
               list_length(Ids, C),
               quantification(N, thing_5), gte(C, N).
```

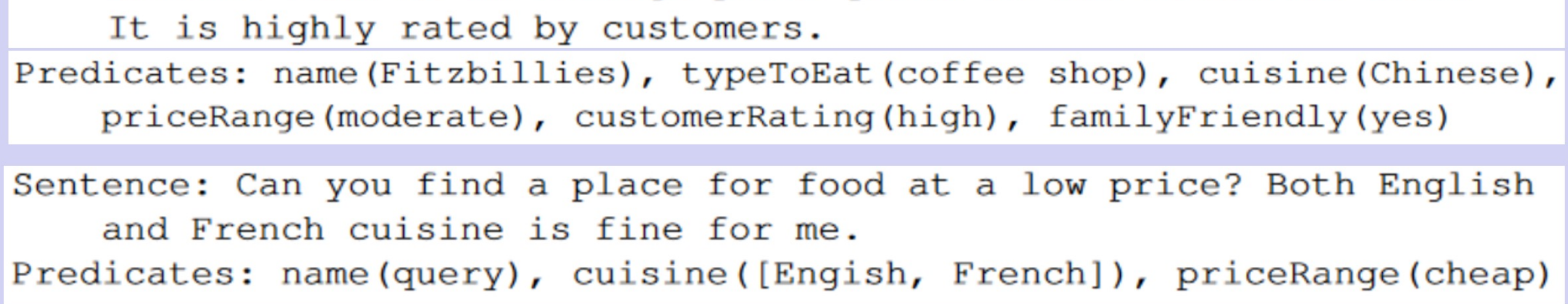
s(CASP) Engine

YES

Basu, Shakerin, & G: PADL 2020

STAR Framework: Trustworthy AI with LLMs

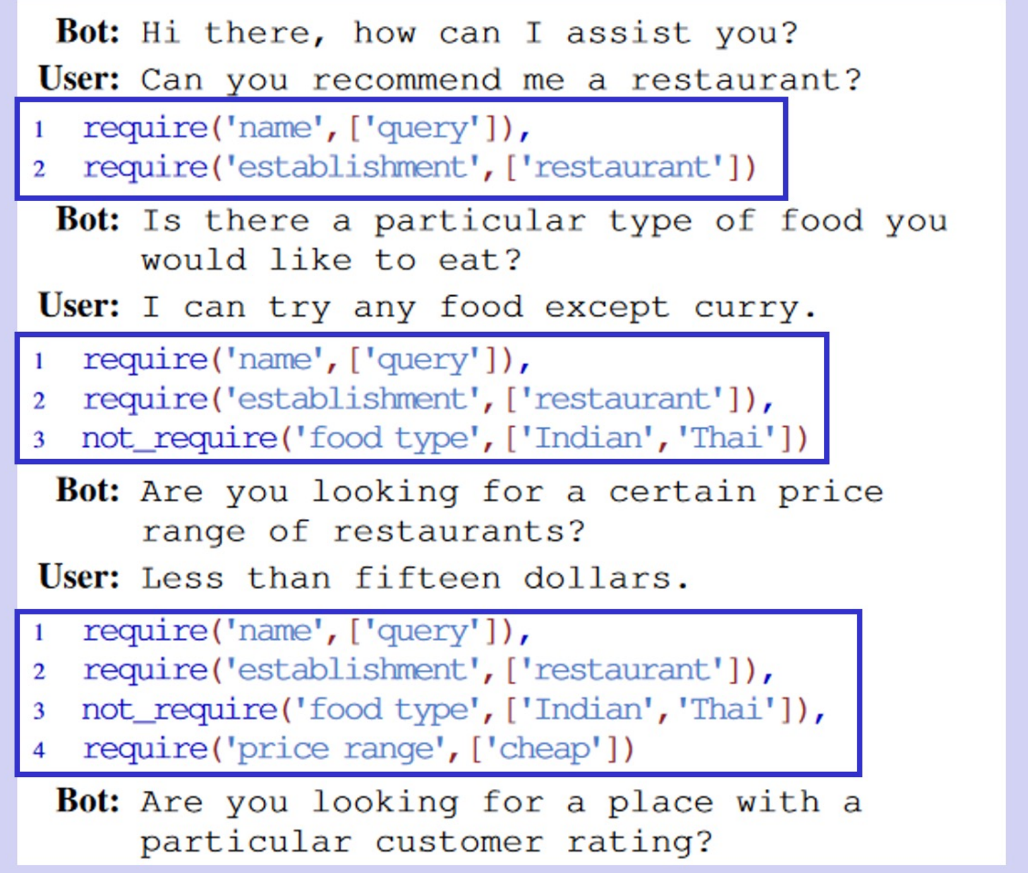
- Think of an LLM as a semantic parser: use it to translate sentences into predicates
- The predicates represent the common vocabulary set by the the user and the AI system;
- The backend AI system implements the intelligent behaviour we want
- The vocabulary is essentially an ontology used to capture a sentence's deep structure
- Turns out that very small number of examples can help LLMs extract knowledge from a sentence relating to a narrow domain; knowledge represented as predicates



```
Sentence: Fitzbillies coffee shop provides a kid-friendly venue for
    Chinese food at an average price point in the riverside area.
    It is highly rated by customers.
Predicates: name(Fitzbillies), typeToEat(coffee shop), cuisine(Chinese),
    priceRange(moderate), customerRating(high), familyFriendly(yes)

Sentence: Can you find a place for food at a low price? Both English
    and French cuisine is fine for me.
Predicates: name(query), cuisine([Engish, French]), priceRange(cheap)
```

Chatbots that "understand"



Bot: Hi there, how can I assist you?

User: Can you recommend me a restaurant?

```
require('name',['query']),
require('establishment',['restaurant'])
```

Bot: Is there a particular type of food you would like to eat?

User: I can try any food except curry.

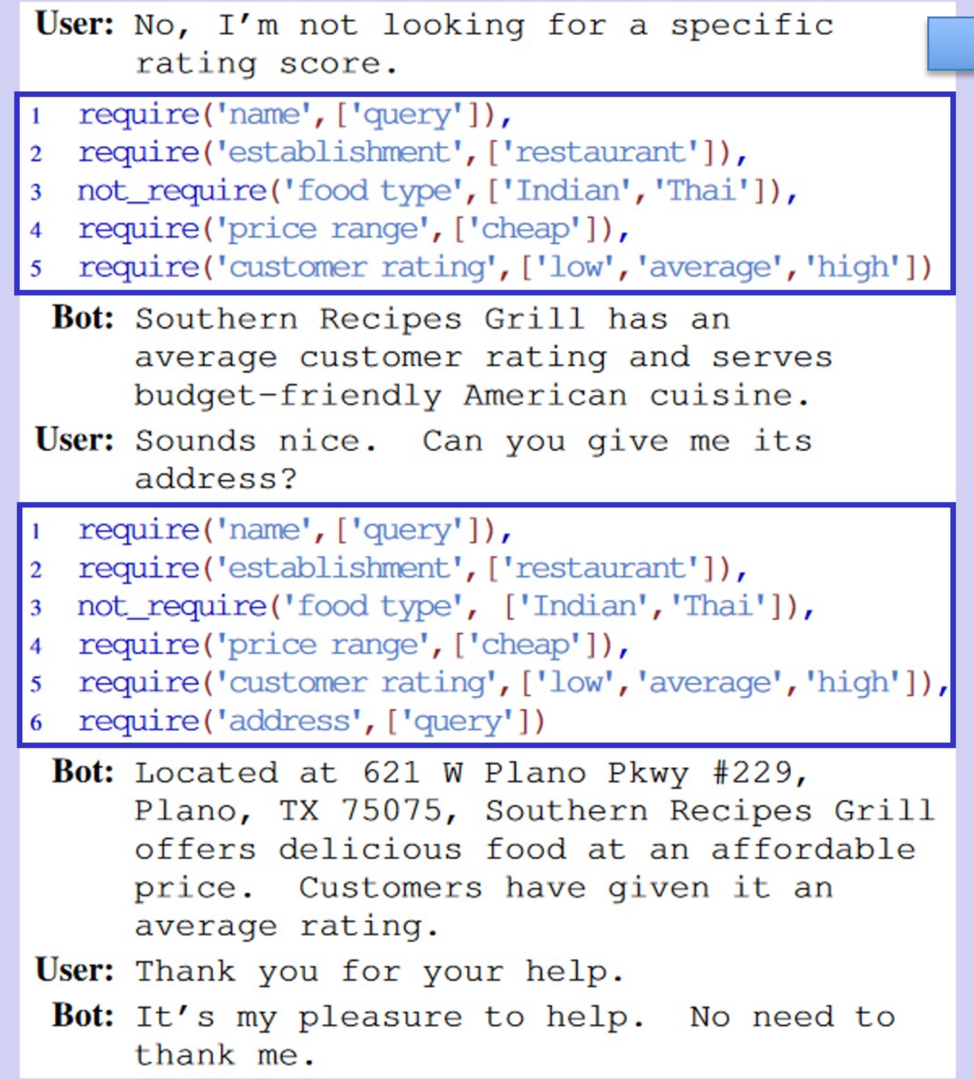
```
require('name',['query']),
require('establishment',['restaurant']),
not_require('food type',['Indian','Thai'])
```

Bot: Are you looking for a certain price range of restaurants?

User: Less than fifteen dollars.

```
require('name',['query']),
require('establishment',['restaurant']),
not_require('food type',['Indian','Thai']),
require('price range',['cheap'])
```

Bot: Are you looking for a place with a particular customer rating?



User: No, I'm not looking for a specific rating score.

```
require('name',['query']),
require('establishment',['restaurant']),
not_require('food type',['Indian','Thai']),
require('price range',['cheap']),
require('customer rating',['low','average','high'])
```

Bot: Southern Recipes Grill has an average customer rating and serves budget-friendly American cuisine.

User: Sounds nice. Can you give me its address?

```
require('name',['query']),
require('establishment',['restaurant']),
not_require('food type', ['Indian','Thai']),
require('price range',['cheap']),
require('customer rating',['low','average','high']),
require('address',['query'])
```

Bot: Located at 621 W Plano Pkwy #229, Plano, TX 75075, Southern Recipes Grill offers delicious food at an affordable price. Customers have given it an average rating.

User: Thank you for your help.

Bot: It's my pleasure to help. No need to thank me.

Can LLMs do it all?

- Can machine learning/LLMs equal an "unerring" human?
  - **No.** LLMs lack "awareness", they don't know what they know and don't know
- Look at nature: sensing == pattern recognition
  - Animals can see objects, recognize sound, recognize touch/smell, and then act on them
  - Dogs have a far better sense of smell than humans, eagles have better eyesight, etc.
  - Animal brain is wired for pattern recognition induced action through evolution
  - Instinctive behavior: pattern-recognition based behavior (a bee will instinctively sting)
  - Wrt pattern recognition: humans and animals more or less have equal capability
- Do animals have reasoning capabilities?
  - As we move up the evolutionary chain, capability to reason improves; humans are the best
- The point: Better reasoning capability means higher intelligence
- If machine learning alone was sufficient, nature would have gotten their first & produced a being that operates purely on instinct & supersedes humans

Controlling LLMs with s(CASP)

- Human sensing and reasoning are intertwined:
  - reasoning may drive sensing or prompt us for closer observation via sensing
  - leverage this for more sophisticated prompt engineering of LLMs and to ensure reliability
- Use an existing strategy to assemble a document, for example; use the reasoning to drive the LLM for extracting document components
- Application: assembling an intelligence report for a given claim
  - Use assurance 2.0 rendering in s(CASP): start with the top-level claim
  - Extract the main verb of the claim and find its correct frame in Framenet (may use LLM)
  - Use VerbNet to extract the elements of the frame (Agent, Patient, Means, …)
  - Generate sub-claims for each element using the LLM;
  - generate defeaters for the subclaims using LLM, to make the argument more robust
  - Repeat this exercise recursively; repeat this exercise for the counter to top-level claim
  - End product: intelligence report based on an LLM but driven by a reasoning template

THE CENTER FOR APPLIED AI AND MACHINE LEARNING
THE UNIVERSITY OF TEXAS AT DALLAS

https://utdallas.edu/~gupta

Computational Cybersecurity in Compromised Environments
2023 Fall Workshop | September 12-14 | Florida Institute of Technology in Melbourne, Florida