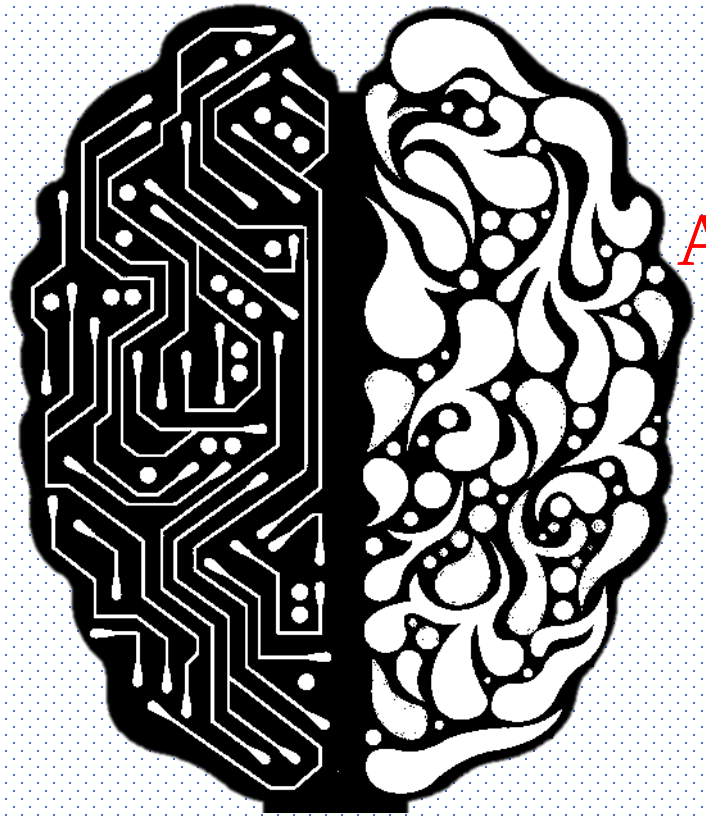


Multi-Agent Systems

Lecture 11

Dr. Nestor eo,

- Researcher CONSUS (Crop OptimisatioN through Sensing, Understanding & viSualisation),
- School of Computer Science
- University College Dublin (UCD)

Shout out to…

Lecture II Learning Objectives

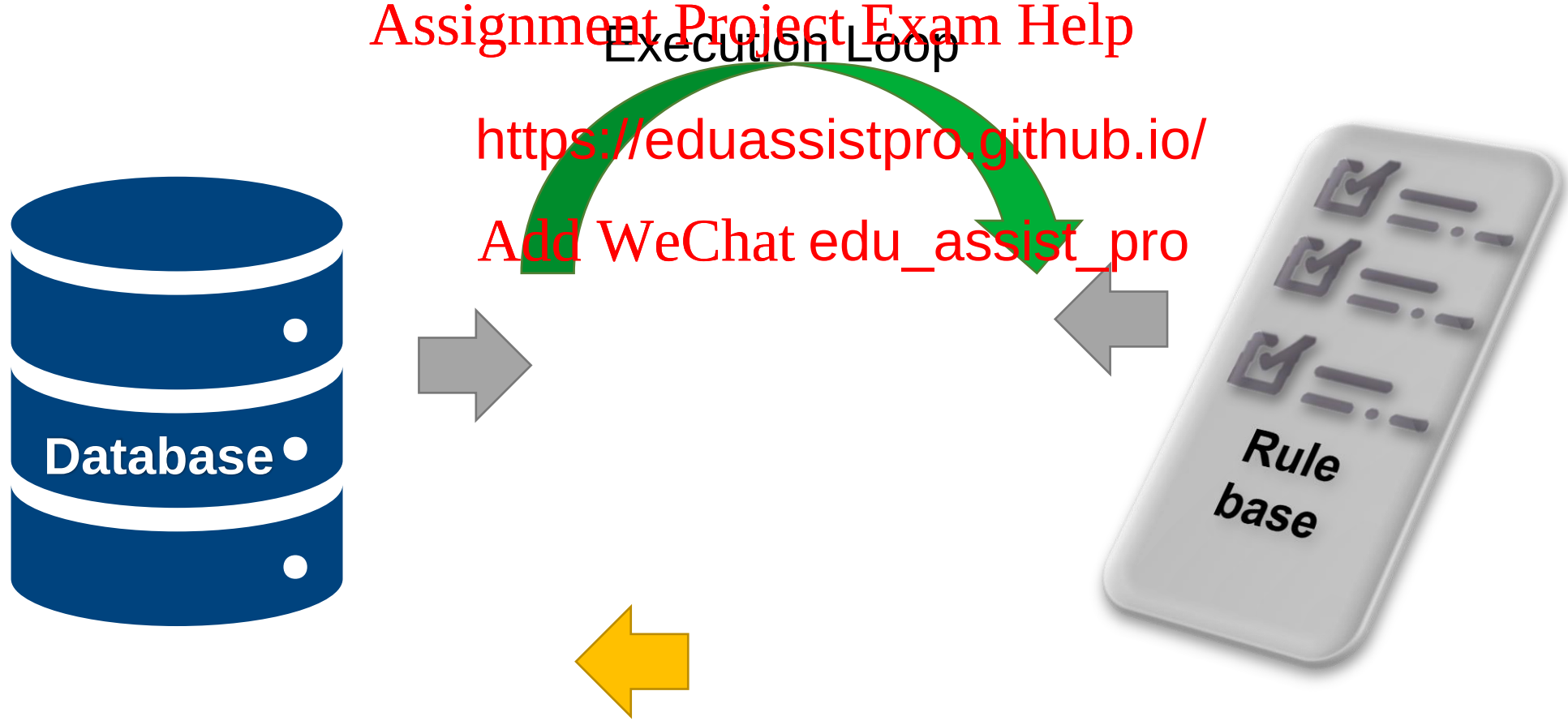
- To understand the elements of an Expert System (ES):
- To understand Inference les of an ES;
- To understand the principles Distributed AI;
- To understand the definition of Agent
- To understand the differences of Agency

Anatomy

rt System

Anatomy of An Expert System

Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** about how **additional state** information can be derived from the current state.



Anatomy of An Expert System

Expert systems start with some initial state relating to a problem domain which they combine with general rules about how additional state information can be derived from the current state.

The data base h l state, typically in the fo bout the problem domain:

e.g. for a weather system, this might be:

LOW_PRESSURE
CLOUDY
COLD

Anatomy of An Expert System

Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** (represent the expertise knowledge as data or rules) about how **additional state** information can be derived from the current state.

The rule base descri
can be derived from existing state.

e.g. for a weather system, this might be:

IF LOW_PRESSURE & CLOUDY
THEN RAIN_LIKELY

IF HIGH_PRESSURE & NOT CLOUDY
THEN RAIN_UNLIKELY

Anatomy of An Expert System

Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** (represent the expertise knowledge as data or rules) about how **additional state** information can be derived from the current state.

Rules are known as **pr**

ANTECEDENT >------> CONSEQ

e.g.

HOT & SUNNY >------> GOOD DAY

The rule base is typically **ordered**.

Anatomy of An Expert System

Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** (represent the expertise knowledge as data or rules) about how **additional state** information can be derived from the current state.

Some systems include

ANTECEDENT >---X---> CONSE

e.g.

HOT & SUNNY >--0.8--> GOOD DAY

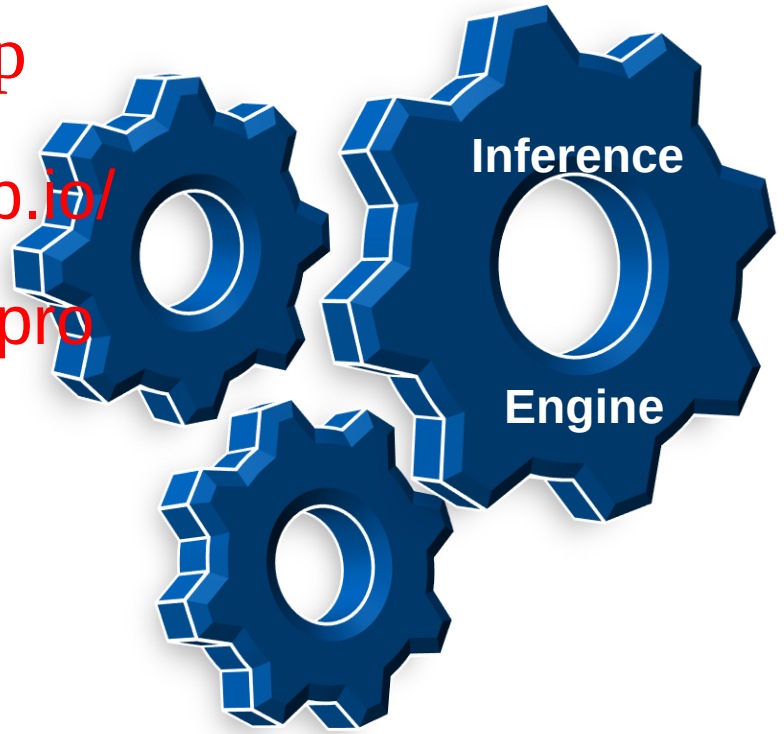
It is 80% certain that it will be a good day if it is hot and sunny.

Anatomy of An Expert System

Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** about how **additional state** information can be derived from the current state.

The inference engine is the procedural part that actually applies th additional state:

1) **Forwards Chaining** inference eng generate all the consequences of the initial state.
2) **Backwards Chaining** inference engines are query oriented – i.e. based on the initial state and rule base, is the following fact true?



Anatomy of An Expert System

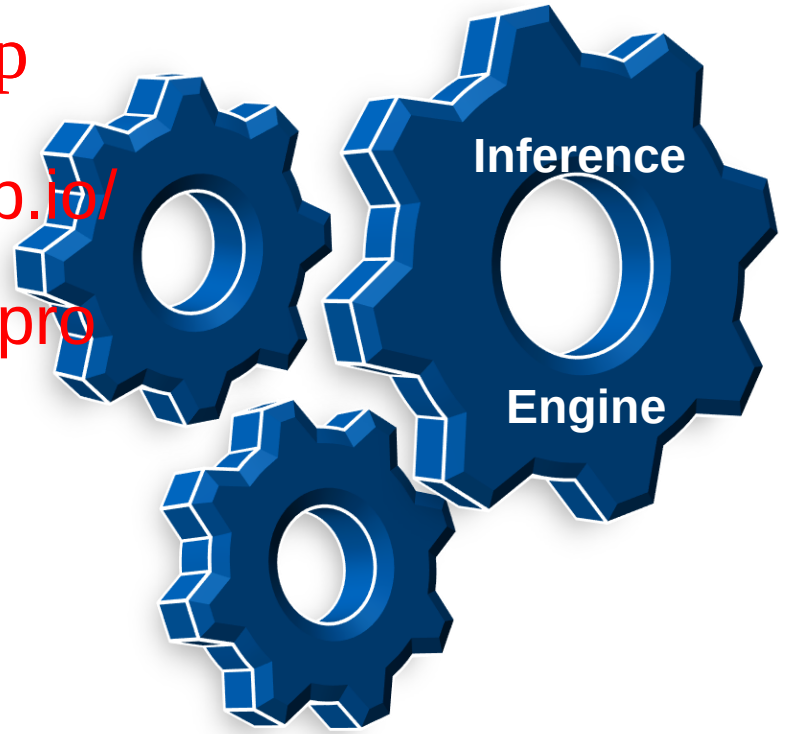
- Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** about how **additional state** information can be derived from the current state.

Forwards Chaining:

The inference engin
a rule and *updates* the database.

If the update does not add new state, the rule is ignored until an update occurs.

If none of the rules add new state, then the inference engine terminates.



Anatomy of An Expert System

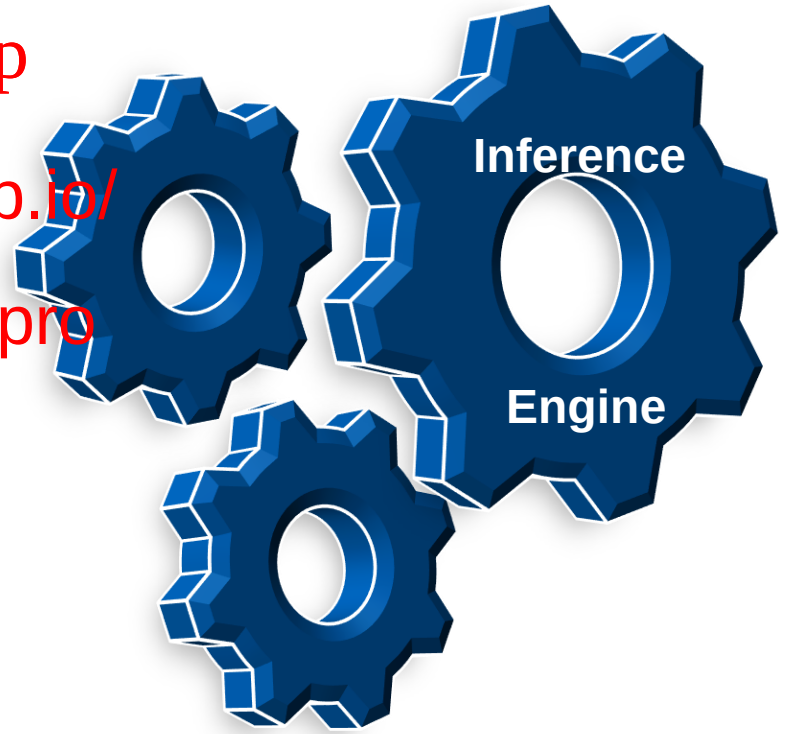
- Expert systems start with some **initial state** relating to a problem domain which they combine with **general rules** about how **additional state** information can be derived from the current state.

Forwards Chaining:

LOW_PRESSU
CLOUDY
COLD
RAIN_LIKELY

IF LOW_PRESSURE & CLOUDY
THEN RAIN_LIKELY

IF HIGH_PRESSURE & NOT CLOUDY
THEN RAIN_UNLIKELY



Anatomy of An Expert System

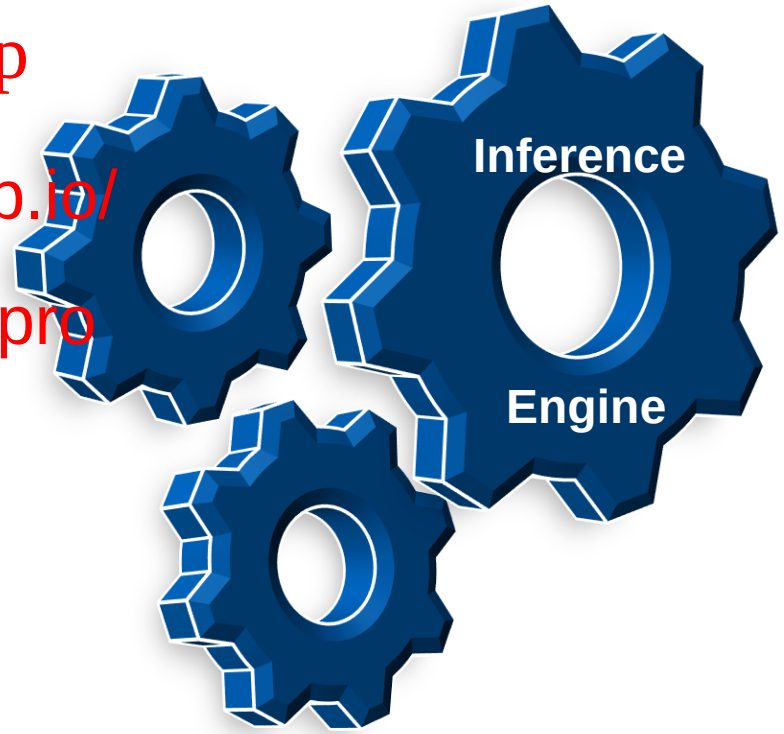
Backwards Chaining:

Start with a question – given the initial state and the rules, is the X true?

Check the data base – if X is th

Check for a rule R where X is a consequent – i is no R then X is false.

Recursively check is the antecedents of R are true.



Anatomy of An Expert System

Backwards Chaining:

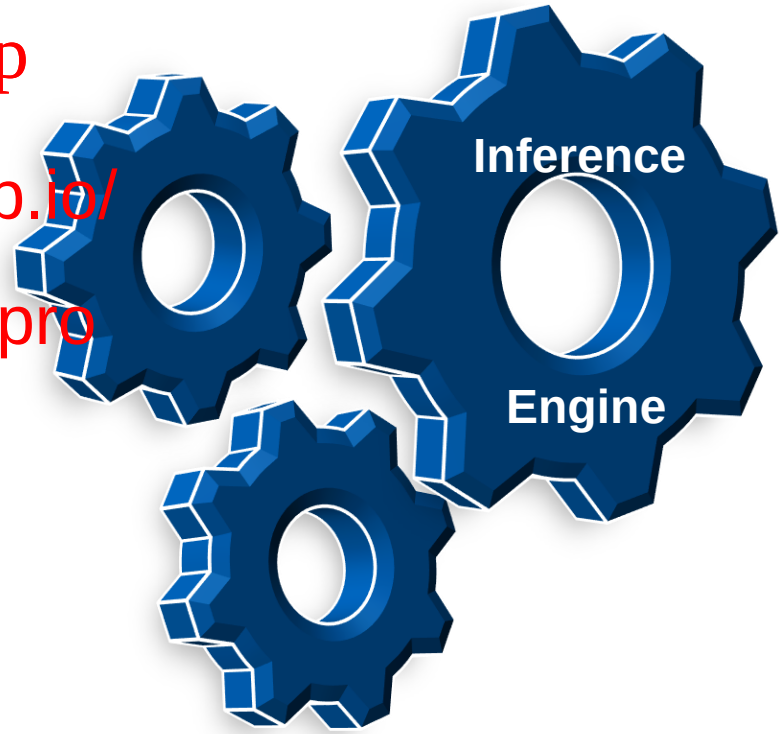
- .

LOW_PRESSURE
CLOUDY
COLD

IF LOW_PRESSURE & CLOUDY
THEN RAIN_LIKELY

IF HIGH_PRESSURE & NOT CLOUDY
THEN RAIN_UNLIKELY

Is RAIN_LIKELY true?



Anatomy of An Expert System

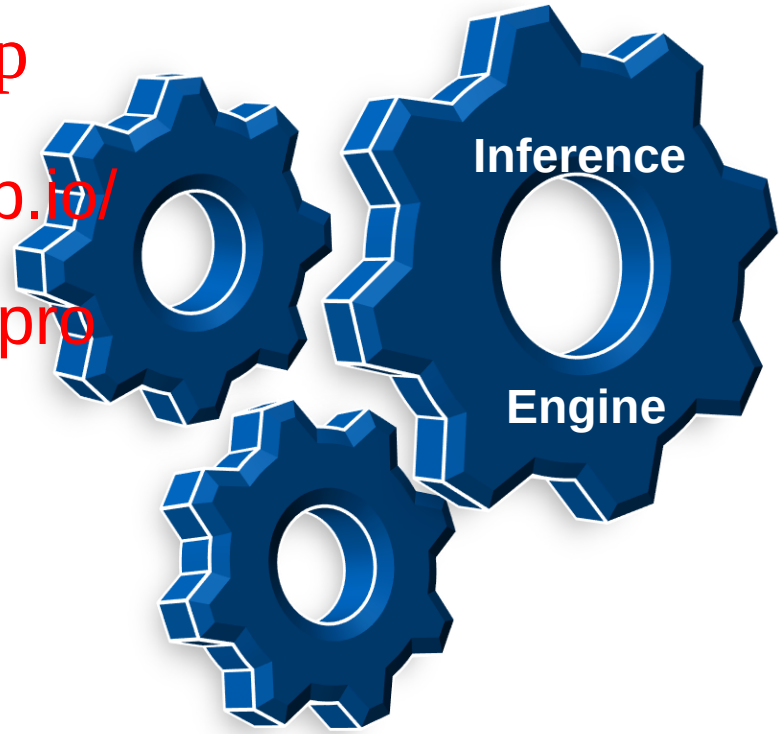
Backwards Chaining:

LOW_PRESSURE
CLOUDY
COLD

IF LOW_PRESSURE & CLOUDY
THEN RAIN_LIKELY

IF HIGH_PRESSURE & NOT CLOUDY
THEN RAIN_UNLIKELY

Is LOW_PRESSURE & CLOUDY true?



Anatomy of An Expert System

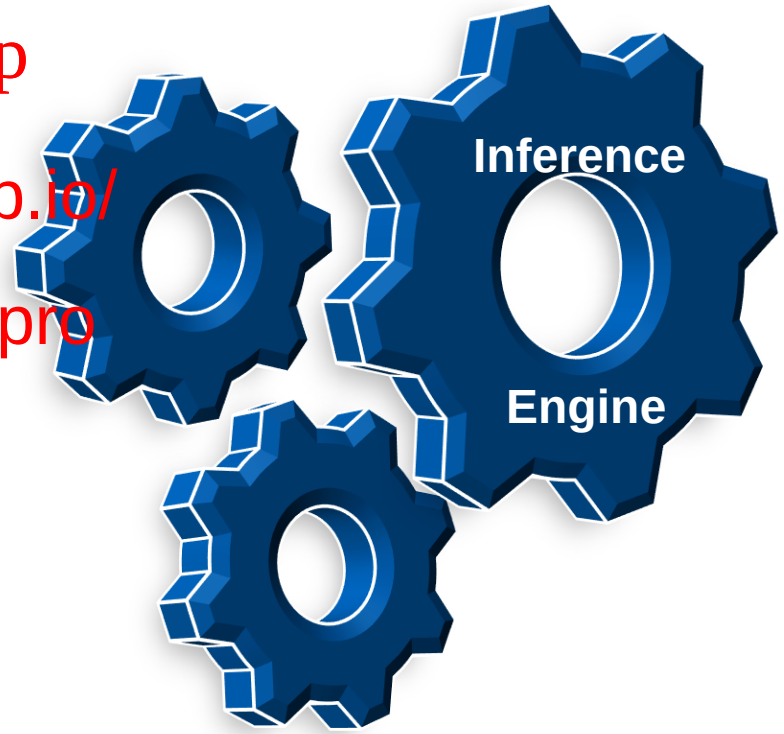
Backwards Chaining:

LOW_PRESSURE
CLOUDY
COLD

IF LOW_PRESSURE & CLOUD
THEN RAIN_LIKELY

IF HIGH_PRESSURE & NOT CLOUDY
THEN RAIN_UNLIKELY

It then follows that RAIN_LIKELY is true!



Example Expert System

- The *rule base*:

1 SALTY AND FRIED --> GOOD
2 MEATY AND NO_V
3 MEATY AND VEGG -> HEALTHY
4 COLD AND NOT MEATY --> YU
5 NOT FRIED OR MEATY-->BAD

Forwards Chaining

- Benefits:
  - Good for query intensive applications:
  - Once you have derived all possible facts, querying is low cost (you can check the database many times).
  - Works well with dynamic
  - Rules can be added to atabase can be easily updated due to changes in the sy

- Drawbacks:
  - Excessive overheads:
    Large rule base = lots of derived facts (very slow)
  - Wasted computations:
    Only a small subset of the derived facts may be required for the queries that are made.

Backwards Chaining

Benefits:

- On-demand inference:
  Derived facts are generated when necessary.
- Optimised Performance:
  Only the pertinent facts

Drawbacks:

- Replication of reasoning:
  Sometimes the same fact may be derived many times for the same state (can be alleviated through caching)
- Loss of intermediate facts:
  Often, any fact derived which checking a query is thrown away once the query is complete.

Beyond Propositional Symbols…

Experts systems can be extended to first-order logic:

- **Facts**: predicates
- **Rules**: Inferences
- **Inference Engine**: mo rd chaining) or resolution (backward chaining).

Example:

- Facts:
  - Is(greg, man), is(man, human)
  - Is(caroline, woman), is(woman, human)
- Rules:
  - is(X, Y) and is(Y, Z) => is(X, Z)
  - is(X, Z) and is(Y, Z) => same(X, Y)