Rule-based expert systems

Based on Ch 2 (and parts of Ch 9) in Negnevitsky

‘General purpose’ intelligence

Ability to:

- Learn from new situations
- Apply solutions from old problems to new ones
- Communicate about any topic, even if not familiar
- Apply abstract reasoning to a range of specific problems
- Understand how the world works in general

The world knowledge problem

General purpose intelligence requires vast amounts of knowledge about the world.

What do you need to know in order to understand the following sentences?

“Sorry, but the bank was closed. Would you get this round?”

“I dropped the carton. I guess we’re having an omelet!”

Domain-specific intelligence

- Restricted to a particular domain (e.g. migraines, oil prospecting, etc.)
- Knowledge is deep, but not wide (e.g. migraines only, no other medical problems)
- Knowledge is tied directly to problem solving (e.g. diagnosis of migraines, not their social repercussions)

→ Avoids the world knowledge problem, and is much more feasible for implementation.

Procedural vs. propositional

- Most programming languages are *procedural*: first do this, then this, then this…
- Most human expert knowledge is *propositional*: A is true. If A is true, then B is true…
- So, procedural programming languages are not necessarily the most intuitive way to represent domain knowledge.

Quantitative vs. qualitative

Humans tend to represent their knowledge qualitatively, e.g.:

- John is tall, but Jim is taller.

As opposed to quantitative:

- John’s height is 6’5”. Jim’s height is 6’8”.

So, probably a good idea to represent expert knowledge that way.

Heuristics vs. rigid rules

- *Heuristic*: A rule of thumb; a guideline.

Most domain knowledge is in the form of heuristics, rather than rigid rules, e.g.:

- “Frequent coughing might mean bronchitis”

Rather than:

- > 10 coughs per hour = bronchitis

Reasoning + knowledge + facts

Human expertise typically breaks down into:

- Ability to reason
- Knowledge about the domain (e.g. migraines)
- Facts about the particular situation (e.g. this patient's symptoms)

Explanation

A human domain expert is usually able to *explain the reasoning*, e.g.:

Why do you think I have a migraine?

Well, you have frequent, intense pain in the temple area, associated with nausea. Also, you aren't taking any medications that are likely to produce these symptoms.

Rule-based expert systems

Designed to:

- Capture a domain expert's knowledge in propositional, qualitative form
- Separate the knowledge from the reasoning (inference) and from the specific facts
- Allow heuristic reasoning and explanation

Production rules

IF <antecedent>

THEN <consequent>

For example:

IF fish is floating

THEN fish is dead

Alternate syntax:

Fish is floating → fish is dead

Note: looks similar to logical “implies”, but less formal – can’t necessarily apply logical transformations.

Multiple antecedents

Multiple antecedents may be connected by conjunction (AND) and/or disjunction (OR).

IF animal is horse-shaped

AND animal has stripes

THEN animal is zebra

IF animal is hippo

OR animal is lion

THEN animal is dangerous

Multiple consequents

Multiple consequents are possible, and are connected by conjunctions, e.g.

IF tsunami alarm is sounding

AND date is not first-Monday-in-month

THEN

Condition is dangerous AND

Advice is ‘move away from the ocean’

Structure of antecedents and consequents

Object is related to **value** by **operator**. For example:

IF ‘taxable income’ > 13914

THEN ‘medicare levy’ = ‘taxable income’ * 1.5 / 100 [Negnevitsky p 27]

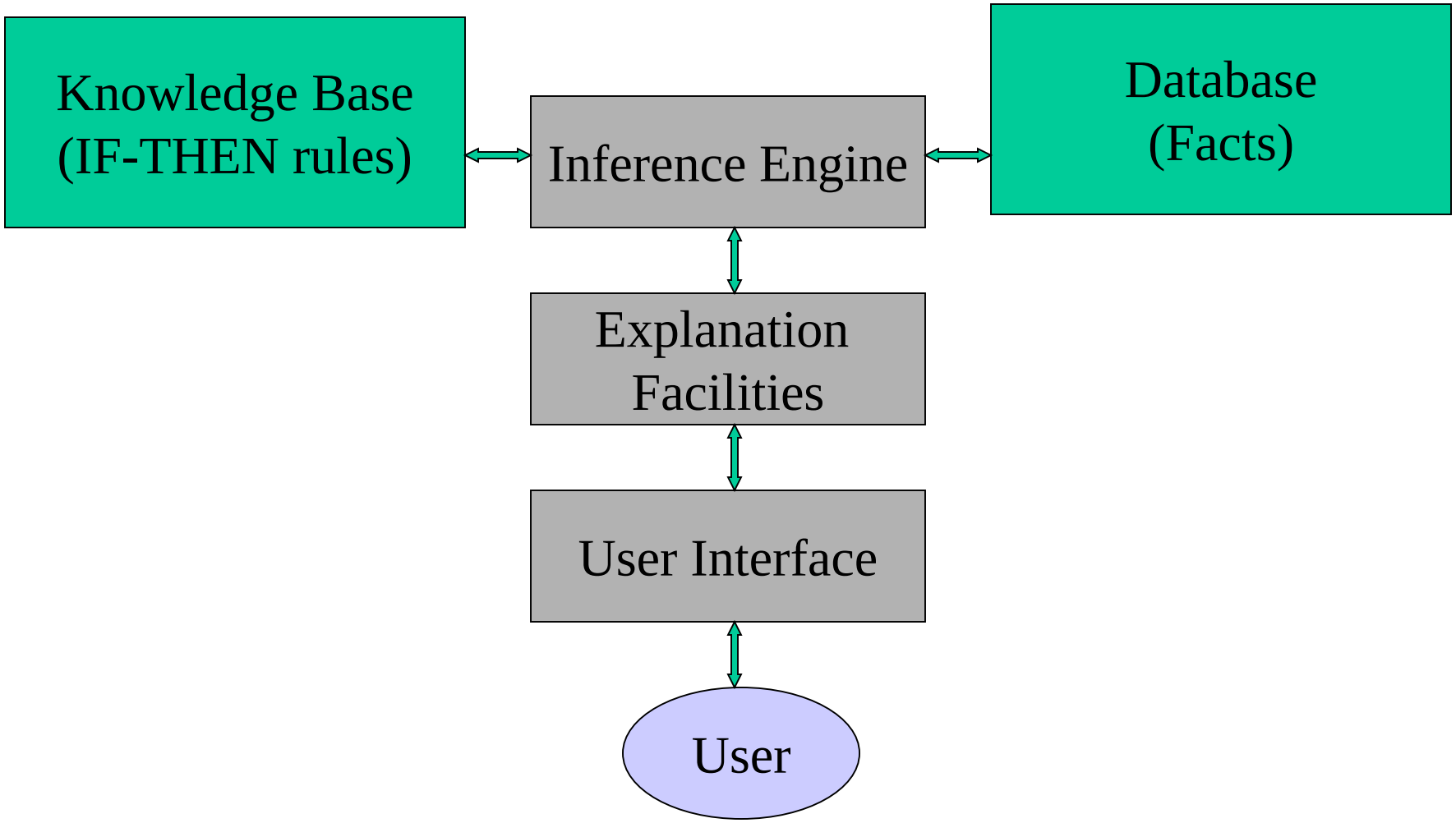
Can get arbitrarily complex, but best to keep it as close to ‘plain English’ as possible.

Semantics of rules

5 main (overlapping) semantics – what's important is that production rules have explicit meaning.

- **Relation:** IF water is falling from sky THEN condition is raining
- **Recommendation**: IF condition is raining THEN advice is 'wear a raincoat'
- **Directive**: IF 'light is red' THEN action is stop
- **Strategy**: IF screen is blank AND power is not checked THEN action is 'check power'; power is checked
- **Heuristic:** IF pain is intense THEN diagnosis is migraine

Structure of a rule-based expert system



Expert system shells

- Have everything except the actual knowledge
- Allow quick development
- May include developer tools, debugging utilities, APIs, knowledge acquisition interface, and other tools.
- Classic example: CLIPS (http://clipsrules.sourceforge.net/) developed at NASA in 1985, now in many different implementations.

Inference engine

- Decides which rules to fire, and in which order
- Asks for new facts from user, as necessary
- Determines when an adequate solution has been found

Forward chaining (data-driven search)

- Starts with known data (facts), or by acquiring data (i.e. asking the user, and storing the results as facts).
- Fire the rules that have an antecedent that matches facts in the database, and add any resulting facts to the database.
- Each rule can fire only once.
- When no more rules can fire, stop.

Forward chaining example

[Negnevitsky, p 37]

- Y & D → Z
- X & B & E → Y
- A → X
- C → L
- L & M → N

Database contains: A, B, C, D, E

Forward chaining +/-

- Good for answering “What is the situation?” kind of questions (e.g. “What kind of animal is this?”)
- Fires a lot of rules, and generates a lot of facts that might be irrelevant to the problem

Backward chaining (goal-driven search)

System has hypothetical solution(s) (e.g. “The patient has type I diabetes”), and tries to prove it.

- Find rules that consequents that match the goal.
- If antecedents match the facts, stop.
- If not, make the antecedents the new *subgoals*, and repeat.

Backward chaining example

[Negnevitsky, p 37]

- Y & D → Z
- X & B & E → Y
- A → X
- C → L
- L & M → N

Database contains: A, B, C, D, E

Backward chaining +/-

- Efficient way to prove or disprove a particular hypothesis.
- *Sometimes* more efficient with a small set of hypotheses
- Can be less efficient than forward chaining if large number of hypotheses

Forward vs. backward chaining

- **Forward chaining** best for analysis and interpretation (e.g. DENDRAL (1971) determines molecular structure of soil sample).
- **Backward chaining** best for diagnosis (e.g. MYCIN (1976) diagnoses infectious blood diseases).

Forward + backward chaining

- Most real expert systems use both
- Primary inference is backward chaining
- Switches to forward chaining when new data is added
- Minimizes pointless queries to user (backward chaining), but exploits any facts that are acquired

Explanation in ESs

- Trace the rules fired in producing the solution (the *inference chain*)
- Tell the user what facts and (high-level) rules were used to find the solution

Problem:

No self-reflection or wider explanation, e.g. can't answer "Why is nausea associated with migraines?"

Conflict sets

A subset of the rules in a knowledge base that can fire at the same time, but have inconsistent consequents. For example:

- X is dog → X is not smart
- X is dog & X has breed = ‘border collie’ → X is smart

Conflict resolution methods (1)

Rules fire one at a time. But which fires first?

- *Order of rules determines order of firing*.
  - Turns declarative knowledge base into procedural program.
  - Difficult to update consistently.
  - Reason for ordering not explicit.
- *Give rules explicit priority*.
  - Can undermine declarative nature if used inappropriately.

Conflict resolution methods (2)

- *Fire the most specific rule (**longest matching strategy**).*
  - Assumes that a specific rule processes more info than a general one. [‘border collie’ rule would fire before more general ‘dog’ rule]
- *Fire the rule based on the data most recently entered*
  - Assumes that recent data is more important than older data.

Pros of rule-based expert systems

- Represent knowledge in near-linguistic, declarative manner that is close to how experts explain their own reasoning.
- Separation of knowledge from processing.
- Some ES approaches are good at handling incomplete or uncertain (next chapter) knowledge.

Cons of rule-based expert systems

- How do the rules relate to each other? Difficult to avoid conflicts in large knowledge bases.
- Unable to learn, and large KBs are difficult to update.
- Rigid behavior – fails very quickly outside of domain of expertise.

Exercise

Write a set of rules, part of a “building maintenance” ES, that would be able to answer the question: “Why did the lights just go out in POST 303D?”.

Break the problem down into subgoals, data-gathering, etc…