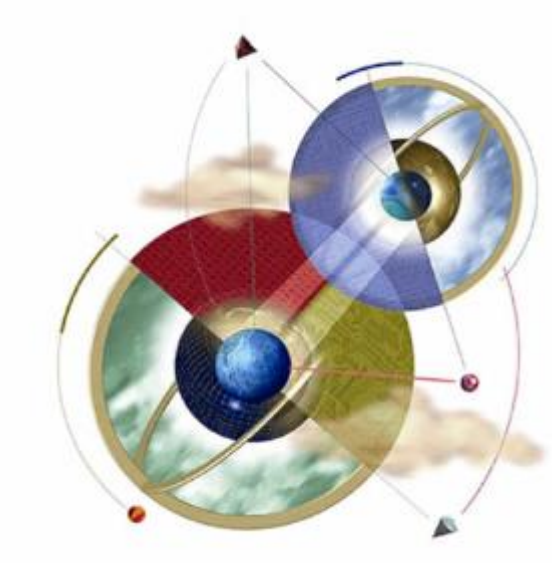


Introduction to Expert Systems

- Introduction
- Main components
- Inference mechanisms

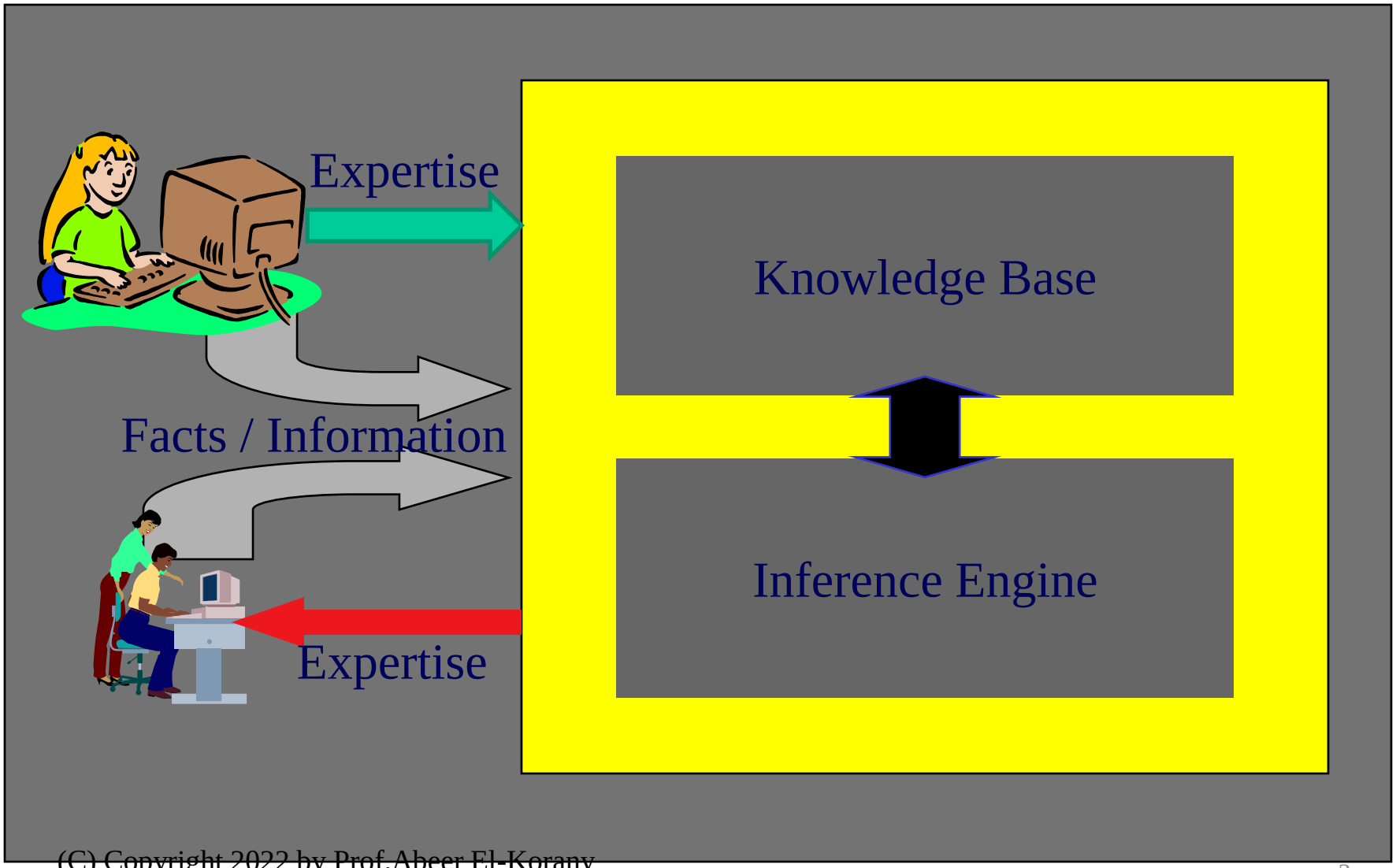
Knowledge Based Systems

The goal is to facilitate **intelligent interaction with the user** based on:

- the identification of the appropriate information
- the effective utilization of the appropriate information
- the control of the appropriate information

in order to **fulfill specific user goals**

Main Components of an KBS



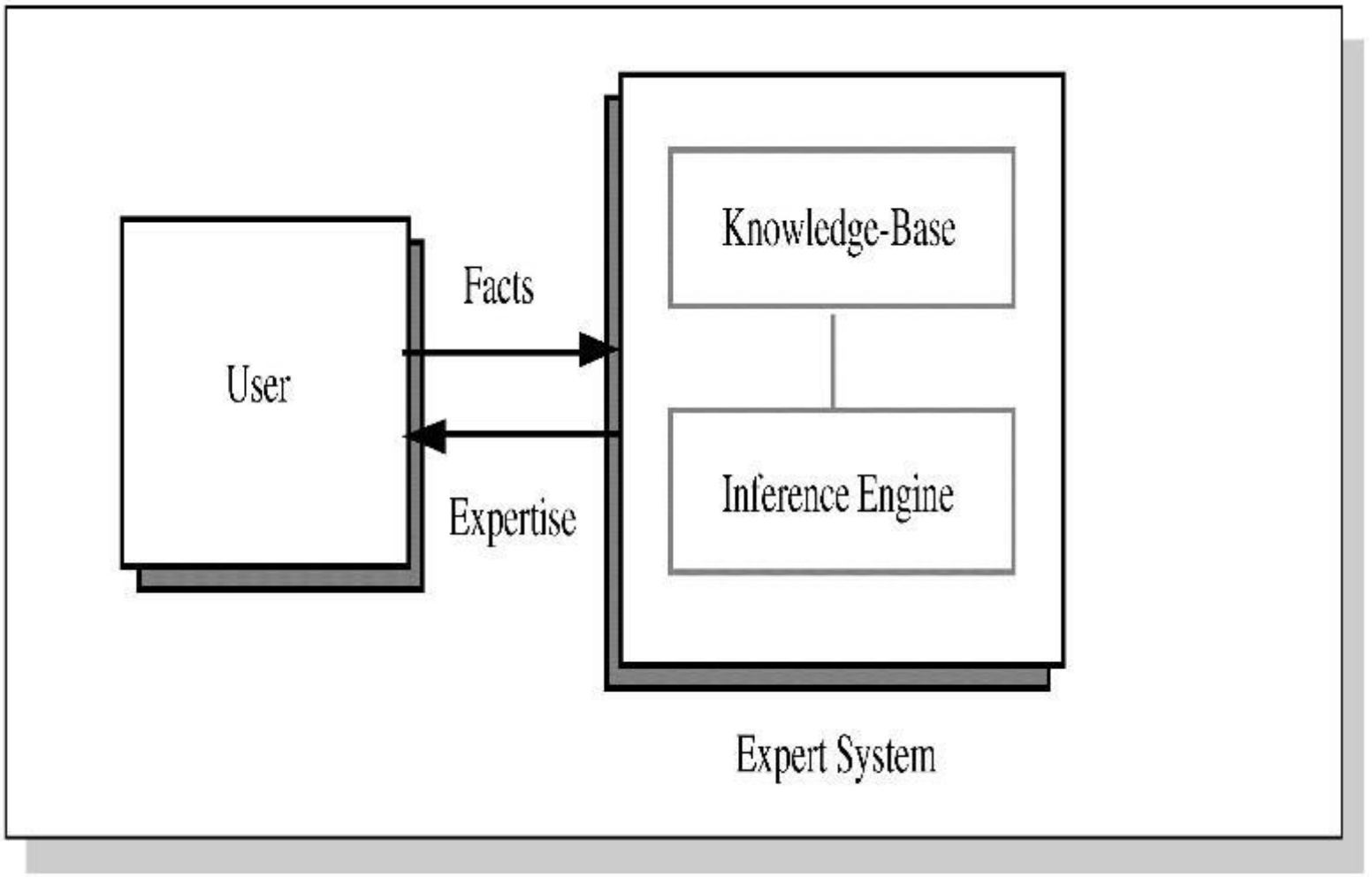
What is an expert system?

"An expert system is a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert."

Professor Edward Feigenbaum

Stanford University

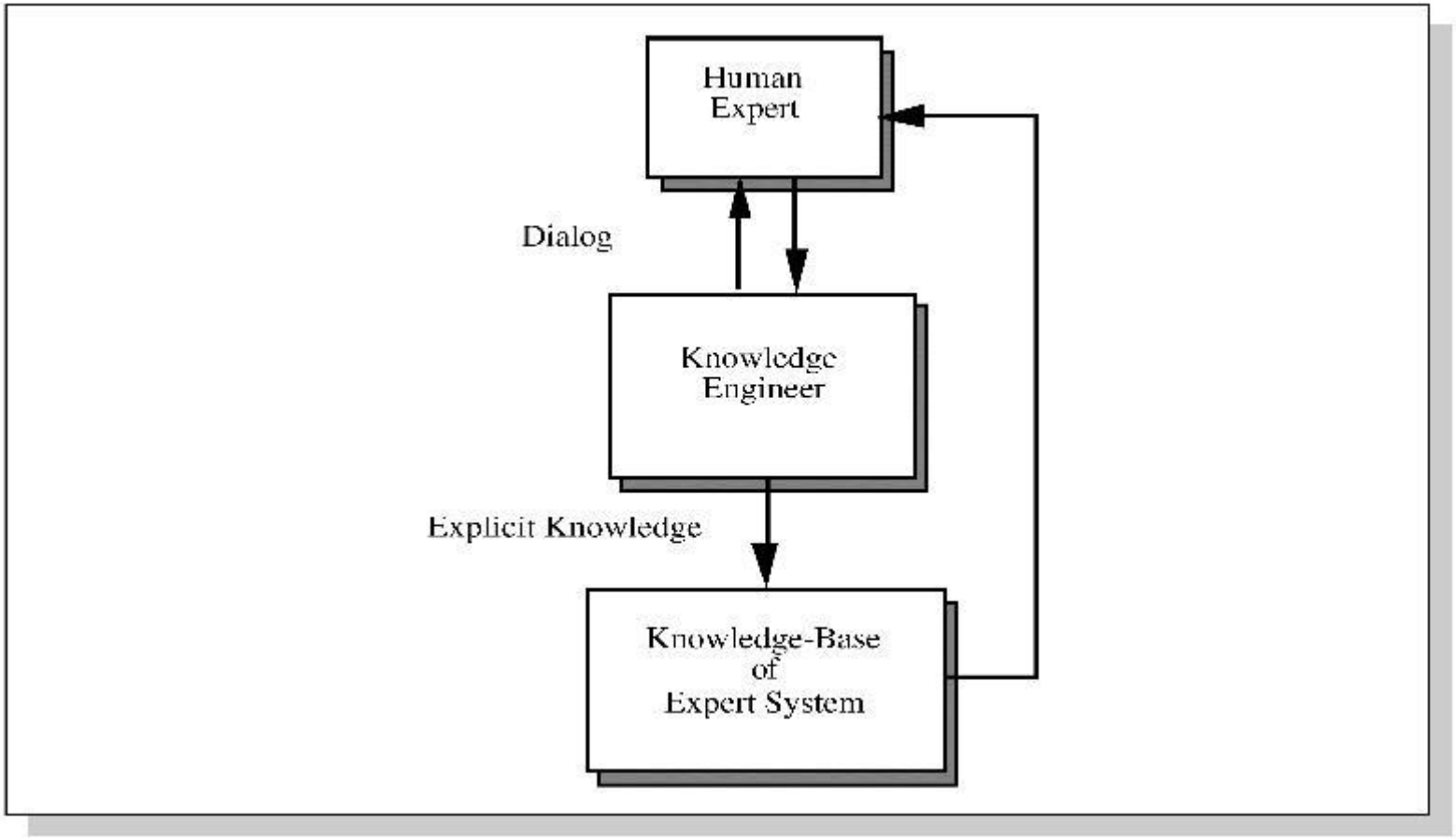
Basic Functions of Expert Systems



Tasks of Expert Systems

Class	General Area
Configuration	Assemble proper components of a system in the proper way.
Diagnosis	Infer underlying problems based on observed evidence.
Instruction	Intelligent teaching so that a student can ask *why*, *how*, and *what if* questions just as if a human were teaching.
Interpretation	Explain observed data.
Monitoring	Compares observed data to expected data to judge performance.
Planning	Devise actions to yield a desired outcome.
Prognosis	Predict the outcome of a given situation.
Remedy	Prescribe treatment for a problem.
Control	Regulate a process. May require interpretation, diagnosis, monitoring, planning, prognosis, and remedies.

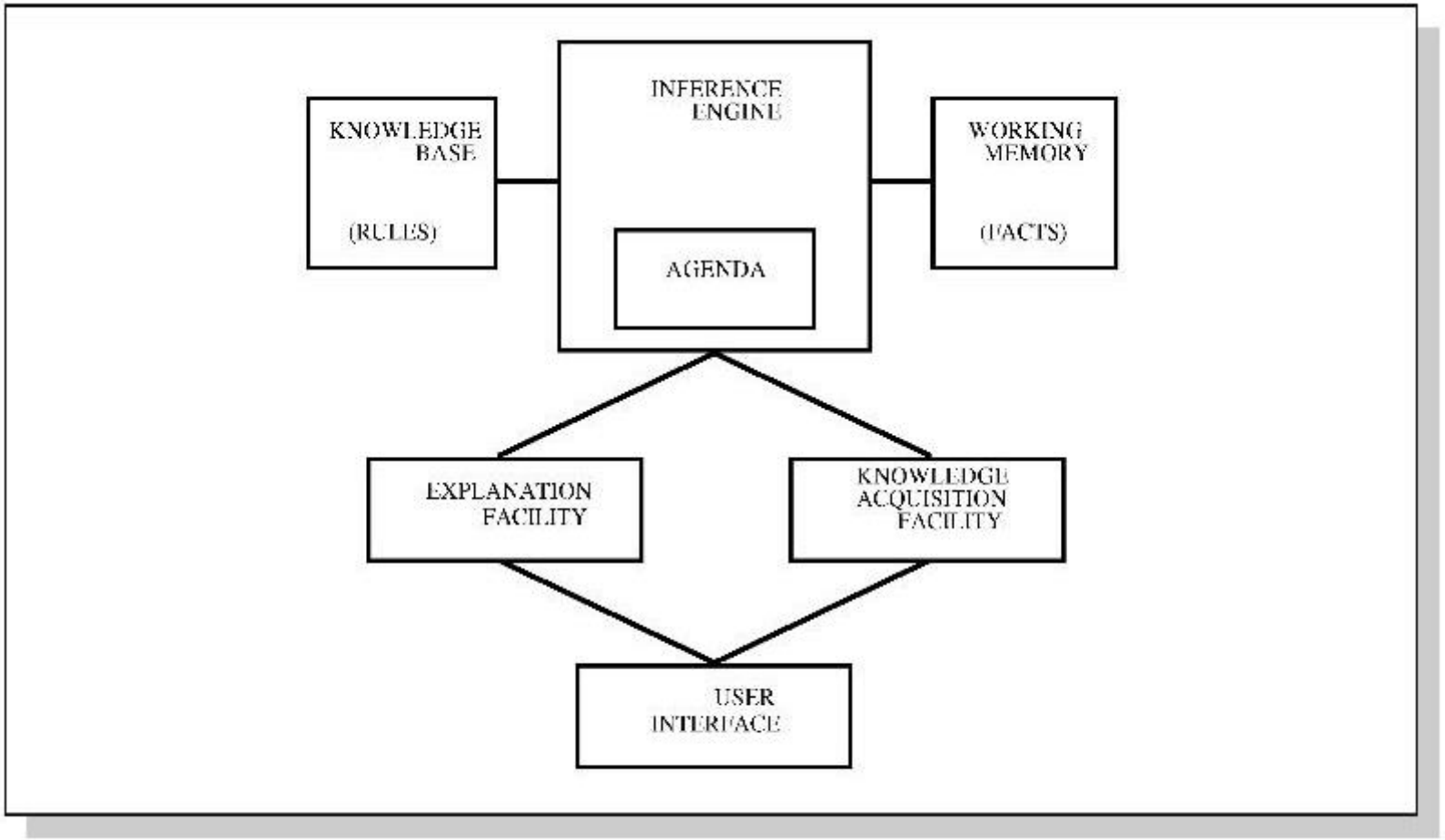
Development of an Expert System



ES Tools

- ES languages
  - higher-level languages specifically designed for knowledge representation and reasoning
- ES shells
  - an ES development tool/environment where the user provides the knowledge base
  - CLIPS, JESS, Mycin, Babylon, G2, ...

Structure of a Rule-Based System



Rule-Based System

- Rules can be used to formulate a theory of human information processing (Newell & Simon)
  - rules are stored in long-term memory (KB)
  - temporary knowledge is kept in short-term memory (agenda)
  - Input or thinking triggers the activation of rules
  - activated rules may trigger further activation
  - a cognitive processor combines evidence from currently active rules
- This model is the basis for the design of many rule-based systems
  - also called *production systems*

- The human mental process is internal, and it is too complex to be represented as an algorithm. However, most experts are capable of expressing their knowledge in the form of **rules** for problem solving.

IF the 'traffic light' is green
THEN the action is go

IF the 'traffic light' is red
THEN the action is stop

Example Rules

IF … THEN Rules

```
Rule: Red_Light
   IF      the light is red
   THEN    stop
Rule: Green_Light
   IF      the light is green
   THEN    go
```

antecedent (left-hand-side)

consequent (right-hand-side)

Production Rules

antecedent (left-hand-side)

```
the light is red  ==>  stop

the light is green  ==>  go
```

consequent (right-hand-side)

Example: Rule-Base to determine the ‘grade’:

1. study → good_grade
2. not_study → bad_grade
3. sun_shines → go_out
4. go_out → not_study
5. stay_home → study
6. bad_weather → stay_home

Q1: If the weather is bad, do you get a good or bad grade?

Rule-Based Systems

- knowledge is encoded as `IF ... THEN` rules
  - these rules can also be written as *production rules*
- the inference engine determines which rule antecedents are satisfied
  - the left-hand side must “match” a fact in the working memory
- satisfied rules are placed on the agenda
- rules on the agenda can be activated (“fired”)
  - an activated rule may generate new facts through its right-hand side
  - the activation of one rule may subsequently cause the activation of other rules

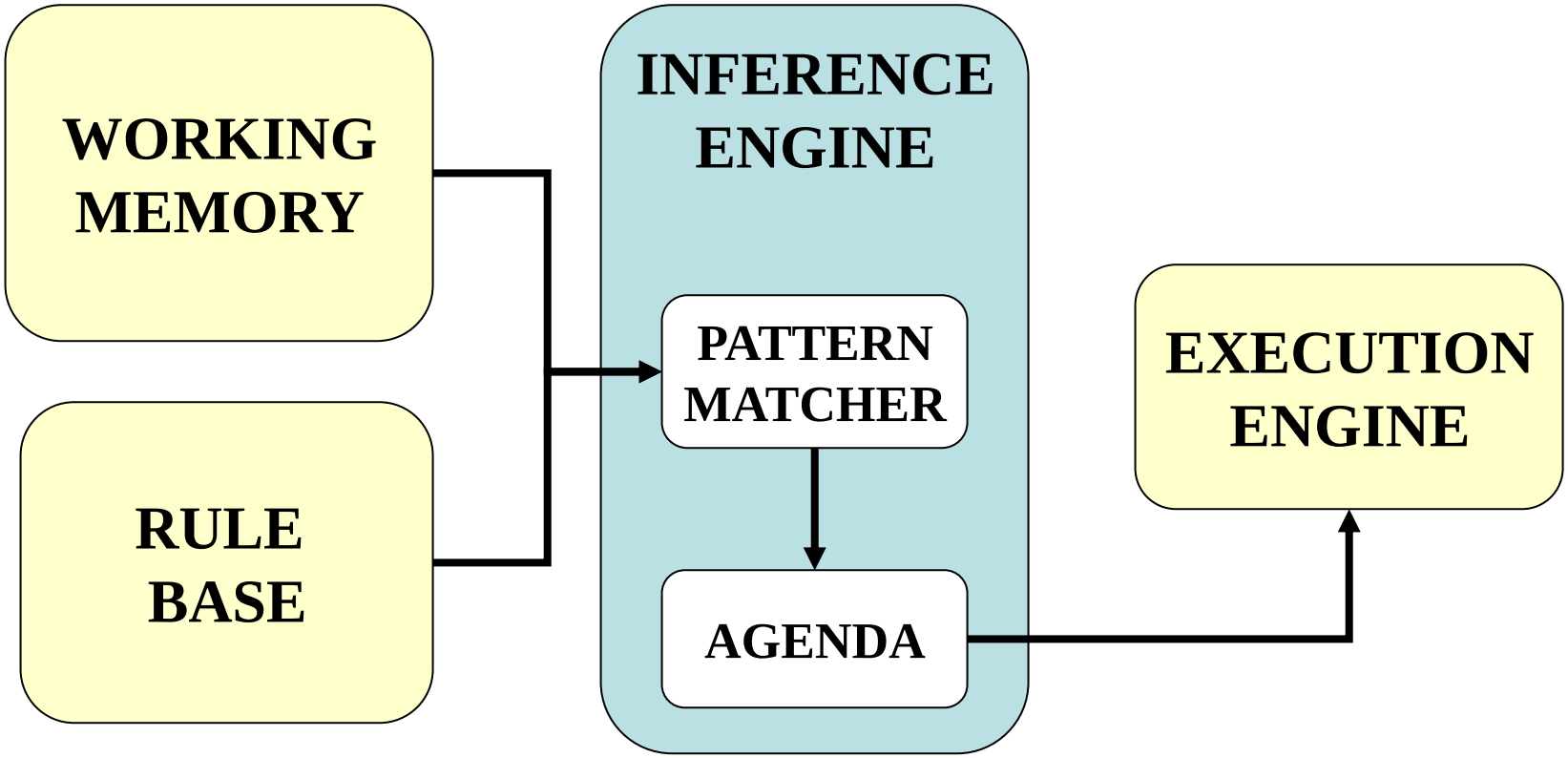
Rule-based reasoning: rules

- The essence of a rule-based reasoning system is that it goes through a series of cycles.

- In each cycle, it attempts to pick an *appropriate* rule from its collection of rules, depending on the present circumstances, and to use it as described above.

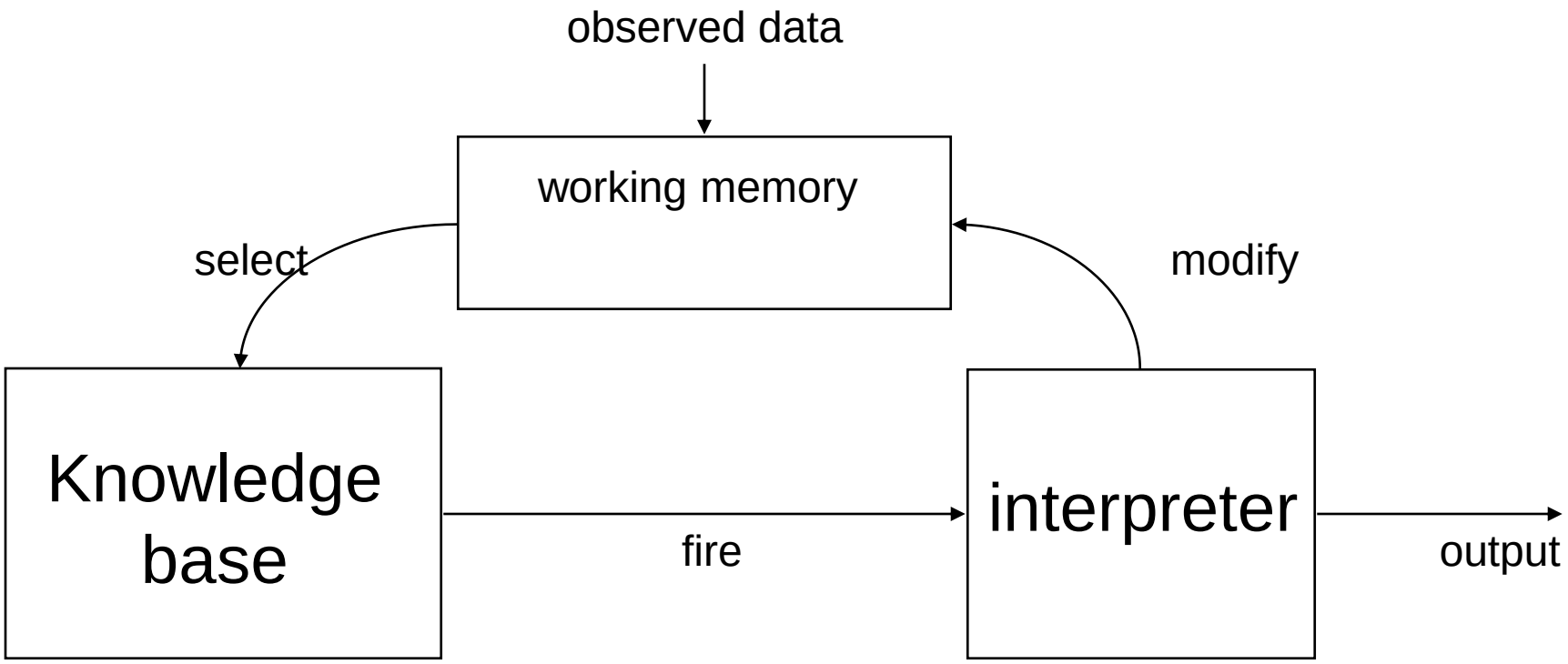
- Because using a rule produces new information, it's possible for each new cycle to take the reasoning process further than the cycle before. This is rather like a human following a chain of ideas in order to come to a conclusion.

Reasoning with production rules



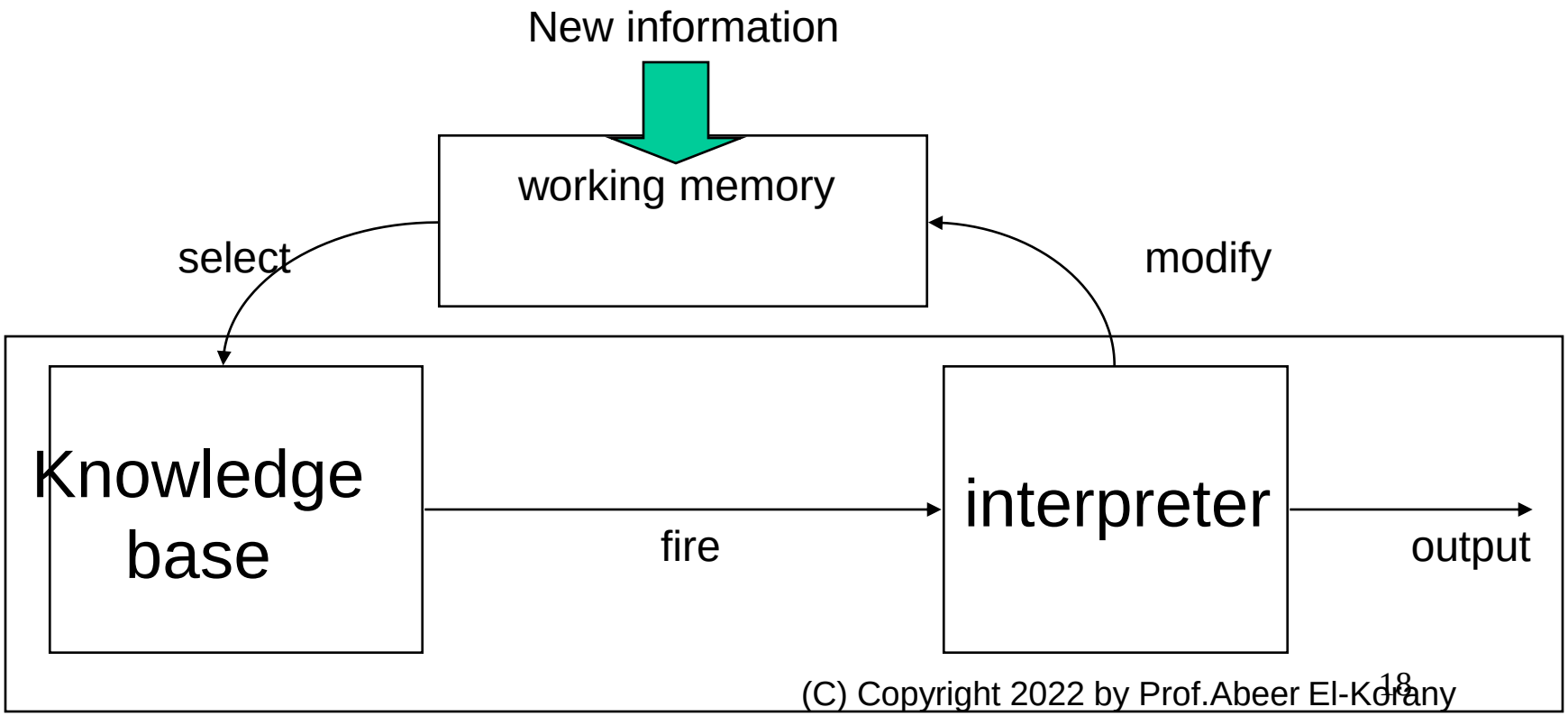
Reasoning with production rules(cont.)

- Architecture of a typical production system:



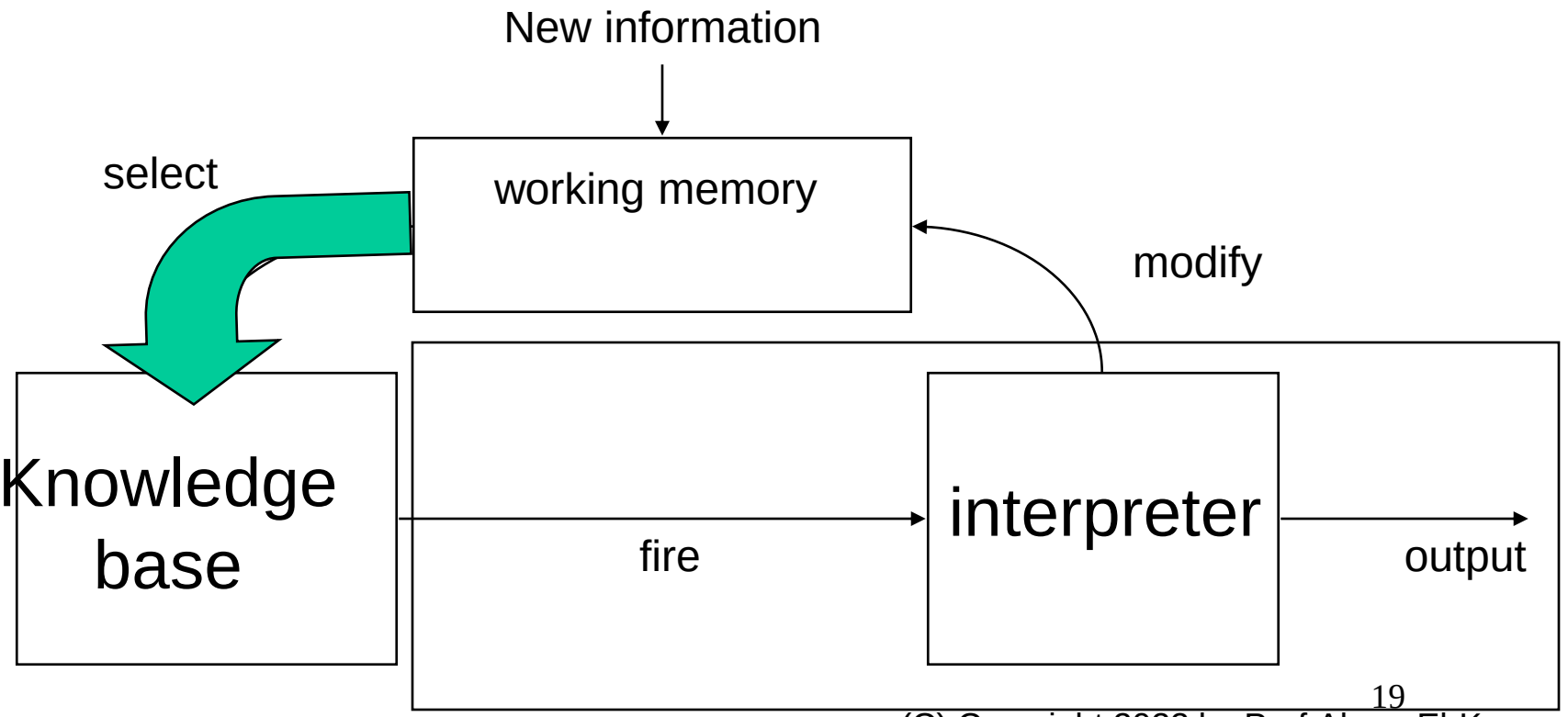
Reasoning with production rules(cont.)

- Architecture of a typical production system:



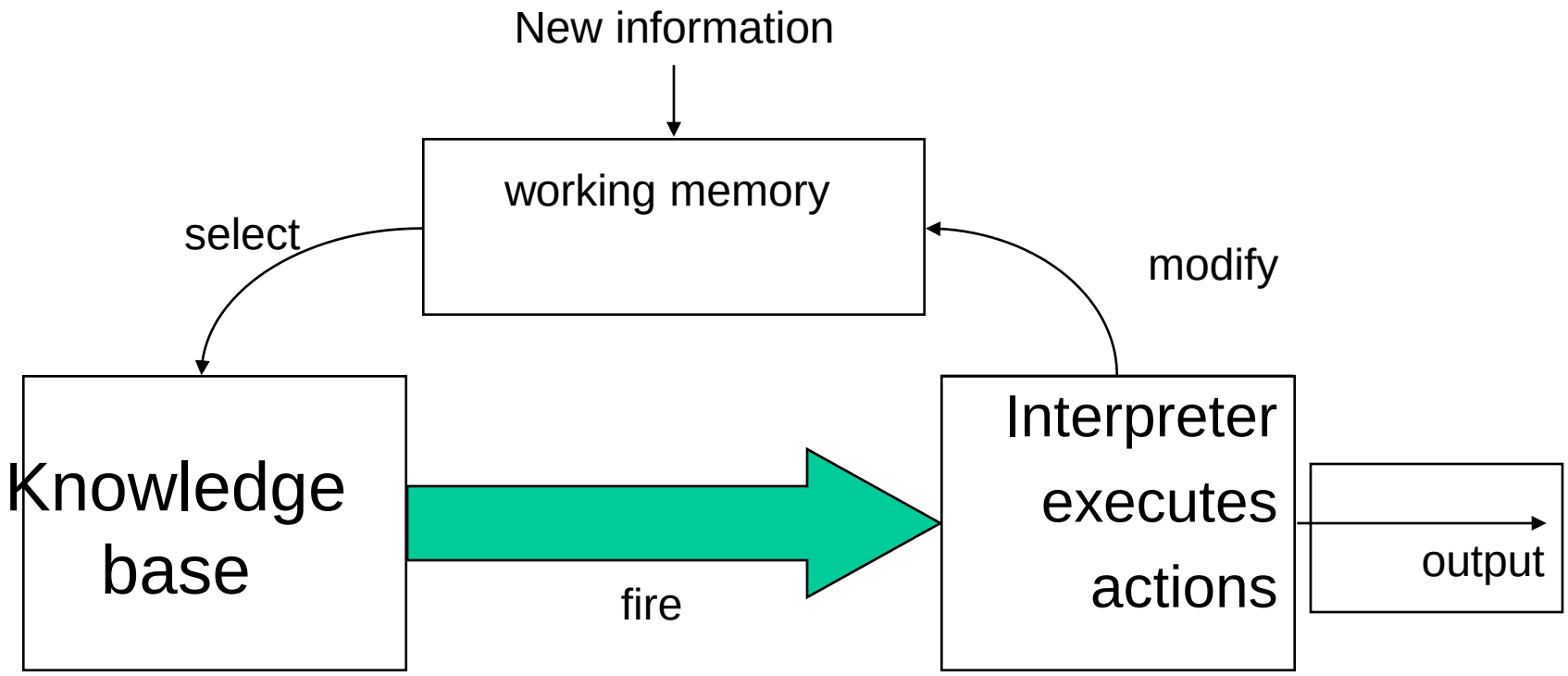
Reasoning with production rules(cont.)

- Architecture of a typical production system:



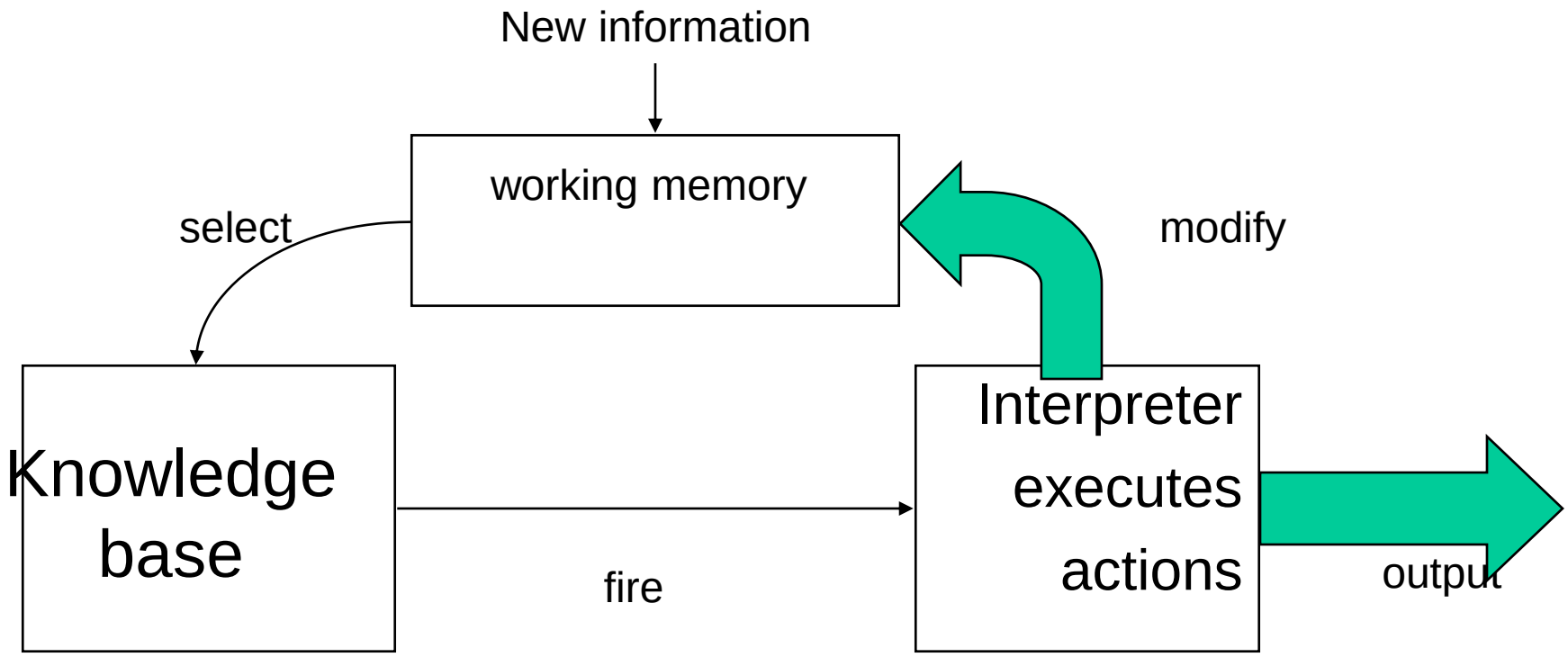
Reasoning with production rules(cont.)

- Architecture of a typical production system:



Reasoning with production rules(cont.)

- Architecture of a typical production system:



Inference Engine Cycle

- Describes the execution of rules by the inference engine
  - match
    - Find rules whose antecedents are satisfied
    - Add them to agenda
  - conflict resolution
    - select the rule with the highest priority from the agenda
  - execution
    - perform the actions on the consequent of the selected rule
    - Update the agenda (remove the fired rule from the agenda and add others)

- The cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

Chaining

- Chain – a group of multiple inferences that connect a problem with its solution
- A chain that is searched / traversed from a problem to its solution is called a forward chain.
- A chain traversed from a hypothesis back to the facts that support the hypothesis is a backward chain.
- Problem with backward chaining is find a chain linking the evidence to the hypothesis.

Control schemes

Two kinds of control in rule-based systems:

- Forward chaining
- Backward chaining.

- Forward chaining starts with the facts, and sees what rules apply (and hence what should be done) given the facts.

- Backward chaining (much like Prolog) starts with something to find out, and looks for rules that will help in answering it.

Forward and Backward Chaining

- Forward chaining (data-driven)
  - Forward chaining is bottom-up reasoning, i.e. reasoning from facts to goals. Reasoning (moving) from facts to the conclusion
  - As soon as facts are available, they are used to match antecedents of rules
  - This process of cascading triggering of rules is called ‘chaining’ as a chain of rules may be fired.
- Backward chaining (goal/query-driven)
  - Starting from a hypothesis (query-goal, supporting rules and facts are sought until all parts of the antecedent of the hypothesis are satisfied)
  - often used in diagnostic and consultation systems

Forward chaining Example

- Consider the simple fire example

R1: IF hot AND smoky THEN there is a fire
R2: IF alarm_beeps THEN smoky
R3 IF there is a fire THEN switch_on_sprinklers

- Working memory initially contains two facts:

Working memory
alarm_beeps .1
hot .2

Follow the algorithm: First cycle.
Find all rules with satisfied conditions : R2
Choose one: R2
Perform actions: smoky

Working memory
alarm_beeps .1
Hot .2
smoky .3

Forward chaining Example (cont.)

- Second cycle:
  - Find all rules with conditions satisfied : R1
  - Choose one and apply action: there is a fire

.

Working memory

alarm_beeps .1
Hot .2
Smoky .3
there is a fire. .4

Third cycle

Rules with conditions satisfied: R3
apply action: switch_on_sprinklers

Forward chaining

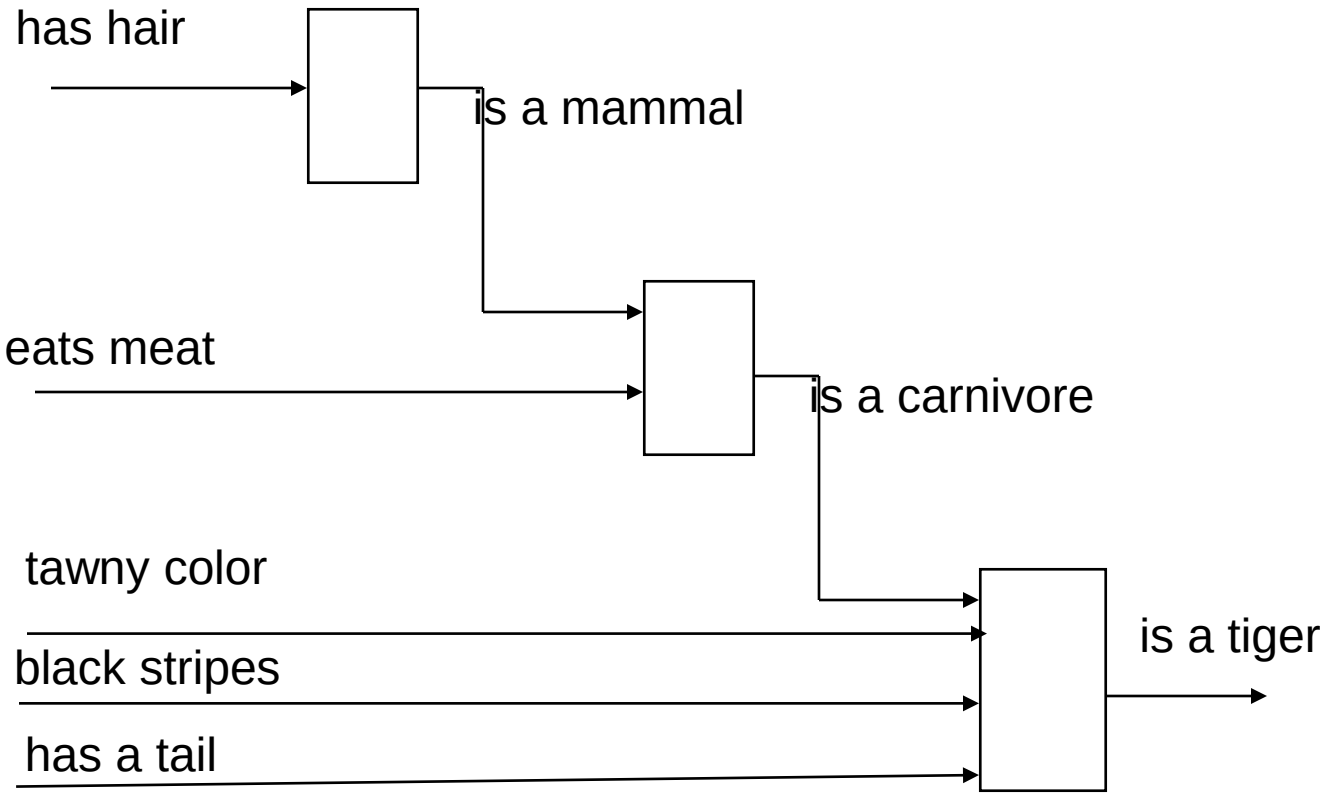
- Facts are held in a *working memory*
- A rule can be activated if all parts of the antecedent are satisfied
- Matches the antecedents of available rules with the current state until it finds a rule all of whose antecedents are satisfied
- If the system supports variables, then matching may require unification
- The inference engine can infer further facts. These new facts may in turn trigger further rules, which may generate further facts.
- Then repeats the cycle
- A rule can be activated if all parts of the antecedent are satisfied
- This defines a **forward-chaining** inference procedure because it moves "forward" from the KB to the goal

Forward Chaining Inferencing (cont.)

1. Scan the rules looking for ones whose premises match the contents of the working memory.
2. Fire the rule which was found.
3. Place its conclusion in the working memory.
4. Until no additional rule fire, go to 1.

Inference Network Representation

- We can represent forward chaining with rules as a network:



Backward Chaining

- Backward chaining is top-down reasoning, i.e. reasoning from goals to facts.

- Basic algorithm, to prove goal G:
  - If G is in the initial facts, it is proven.
  - Otherwise, find a rule which can be used to conclude G, and try to prove each of that rule's conditions.
- Avoid repeated work: check if new subgoal
  - Has already been proved true
  - Has already failed

- This allows rather more focused style of reasoning. (Forward chaining may result in a lot of irrelevant conclusions added to working memory.)

Backward Chaining Example

- Consider the simple fire example
- Following the algorithm: First cycle.
  - Determine the goal
  - Set R3 switch on the sprinklers as G1
- G1: switch_on_sprinklers
  - Is it in initial facts? No.
  - Is there a rule which adds this as a conclusion? Yes, R3
  - Set condition of R3 as new goal

Working memory
alarm_beeps .1
hot .2

- G2: there is fire.
- Second cycle:
  - Is it in initial facts? No.
  - Is there a rule which adds this as a conclusion? Yes, R1

Backward Chaining Example (cont.)

- Set conditions as new goals: G3: hot, G4: smoky.
- G3: hot, G4: smoky
- Third cycle:
  - Try to prove G3: hot. In initial facts.
  - Try to prove G4: smoky. Conclusion of rule R2

- Fourth cycle:
- G5: alarm_beeps.
  - In initial facts, so all done…
  - Proved hypothesis *switch_on_sprinklers*.

Forward &Backward Chaining Example (Cont.)

- A forward chaining system would note first that it knew A and B and could therefore conclude C– then that it knew B and C and could then conclude D– finally note that fortuitously, D was what it wanted to prove

- A backward chaining system would– note first that it needed to prove C and B to prove D– then that it already knew B then that it needed to prove A and B to prove C finally that it already knew A and B.

- Example

  An expert system used to make classification between Ostrich and Duck have the following rules: -

  R1: IF have wing = yes and mammel = yes THEN bird = yes

  R2: IF bird = yes and neck length = long THEN can-fly = no

  R3: IF bird = yes and leg length = small THEN can-fly = no

  R4: IF can-fly = no and leg length = small and neck length = small THEN Duck = yes , Ostrich = no

  R5: IF can-fly = no and leg length = long and neck length = long THEN Duck = no , Ostrich = yes

Trace the algorithm of reasoning with the following user inputs:

[mammel = yes , have wing = yes , leg length = small ,neck length = small]

Apply backward chaining: Goals (Duck Ostrich).

Select the first top goal Duck .

- Rules R4 & R5 that derive that goal placed on the stack .
- Rule R4 examined , the first parameter (can fly =fly) examined to see if it is in the database (which is not) So that makes can fly as sub-goal (and go back to step 2).
- R2 & R3 derive value for (can fly =fly)
- Rule R2 examined , examine-> R1 derive value for (bird)
- Rule R1 examined ,
- Premises of R1 statisfied and get (bird=yes)
- Rule R2 examined ->R2 fails
- Rule R3 examined ->R3 satisfied (fly=no)
- Rule R4 all three premises true then it is satisfied and drive (Duck=yes , Ostrich=no)

Forward Chaining Example

Rules:

R1: IF on-cl(green)
THEN put-on-cl(produce)

R2: IF on-cl(packed in small containers)
THEN put-on-cl(delicacy)

R3: IF on-cl(refrigerated)OR put-on-cl(produce)
THEN put-on-cl(perishable)

R4: IF put-on-cl(perishable)& (weighs=15 lbs) & (inexpensive)
THEN put-on-cl(staple)

R5: IF on-cl(meat) & put-on-cl(perishable) & (weighs=15 lbs)
THEN put-on-cl(turkey)

R6: IF (weighs=15 lbs) &put-on-cl(produce)
THEN put-on-cl(watermelon)

F1- on-cl(green)
F2- weighs =15 lbs

Example - Execution

Fact List

(green, weighs =15 lbs)

Applicable Rules

R1

Example - Execution

Fact List	Applicable Rules
(green, weighs=15 lbs)	R1
(green, weighs=15 lbs, put-on-cl(produce)	~~R1~~, R3, R6

Example - Execution

Fact List	Applicable Rules
(green, weighs =15 lbs)	
(green, weighs =15 lbs, put-on-cl(produce))	~~R1~~, R3, R6
(green, weighs =15 lbs, put-on-cl(produce), perishable)	R3, ~~R6~~

Example - Execution

Fact List	Applicable Rules
(green, weighs = 15 lbs)	R1
(green, weighs =15 lbs, produce)	~~R1~~, R3, R6
(green, weighs =15 lbs, produce, perishable)	~~R1~~, ~~R3~~, R6
(green, weighs =15 lbs, produce, perishable, watermelon)	~~R1~~, ~~R3~~, ~~R6~~ ALL DONE!

Example:Auto Diagnosis

engine-gets-gas — Rule 1: **if** the engine is getting gas,
engine-turns — **and** the engine will turn over,
Goal (problem(spark-plugs)) — **then** the problem is spark plugs.

not(engine-turns) — Rule 2: **if** the engine does not turn over,
not(lights-on) — **and** the lights do not come on,
Goal **(**problem (battery or cables)) . — **Then** (problem(battery-or-cables))

not(engine-turns) — Rule 3: **if** the engine does not turn over,
lights-on — **and** the lights do come on,
Goal(problem(starter-motor)) — **then** the problem is the starter motor.

gas-in-tank — Rule 4: **if** there is gas in the fuel tank,
gas-in-carburettor — **and** there is gas in the carburettor,
engine-gets-gas — **then** the engine is getting gas.

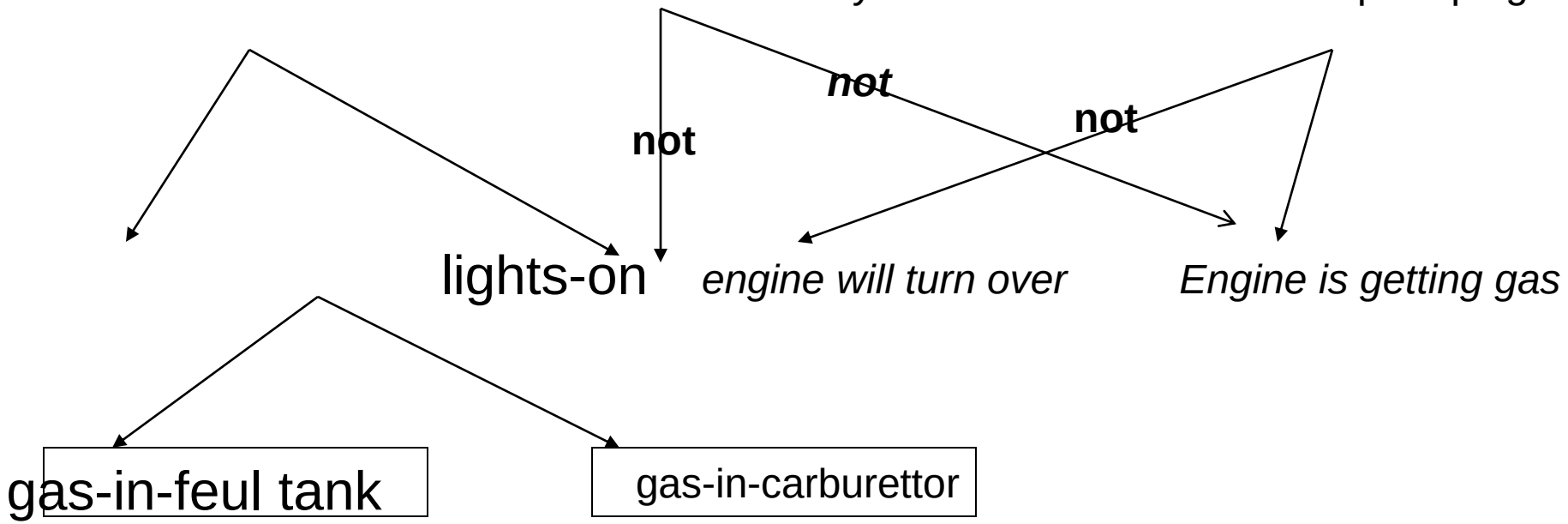
Matching Rules

R4

Working Memory

there is gas in the fuel tank
there is gas in the carburettor
the engine will turn over

Reasoning tree



Simple Medical Expert System

- Rules
  1. If headache then prescribe pain killer
  2. If headache and sore throat and coughing then diagnose flu
  3. If tired and headache then diagnose glandular fever
  4. If tired and sore throat then diagnose tonsillitis
  5. If tired than diagnose stress

Simple Medical Expert System

- User interaction
  - Patient: I have a headache
  - System: I suggest you take a pain killer. Do you have a sore throat?
  - Patient: Yes, I do.
  - System: Have you been coughing?
  - Patient: No,
  - System: Have you been feeling tired recently?
  - Patient: Yes, I have.
  - System: It sounds as though you may have glandular fever or tonsillitis.

Forward vs. Backward Chaining

Forward Chaining	***Backward Chaining***
planning, control	diagnosis
data-driven	goal-driven (hypothesis)
bottom-up reasoning	top-down reasoning
find possible conclusions supported by given facts	find facts that support a given hypothesis
similar to breadth-first search	similar to depth-first search
antecedents (LHS) control evaluation	consequents (RHS) control evaluation

46

Advantages of Rule Systems

- Separation of control (inference) from knowledge
- Modularity of knowledge (A rule is an independent chunk of knowledge)
- Ease of expansion (Proportional growth of intelligence)
- Derivation of explanations from rigid syntax
- Consistency checking
- Utilization of uncertain knowledge
  - IF it looks like rain THEN I should probably carry an umbrella
  - IF Weather looks like rain THEN Carry an umbrella CF 80
- Can incorporate variables

IF ?Student marks is adequate

THEN ?Student can graduate