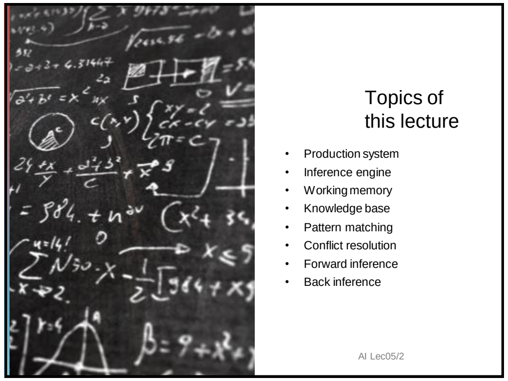
Production System

AI Lec05/1

1

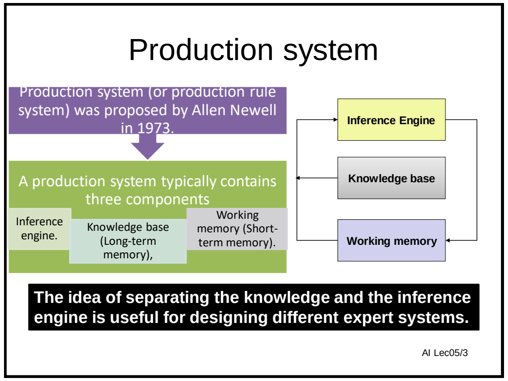


Topics of this lecture

- Production system
- Inference engine
- Working memory
- Knowledge base
- Pattern matching
- Conflict resolution
- Forward inference
- Back inference

AI Lec05/2

2



Production system

Production system (or production rule system) was proposed by Allen Newell in 1973.

A production system typically contains three components

Inference engine.

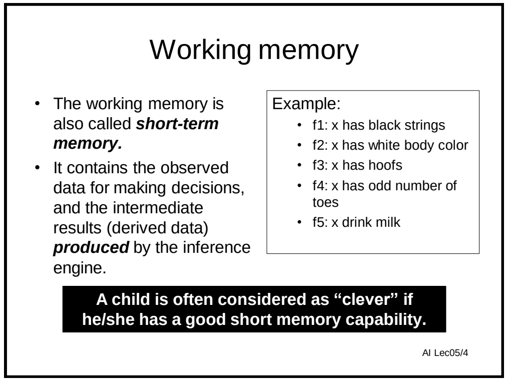
Knowledge base (Long-term memory),

Working memory (Short-term memory).

The idea of separating the knowledge and the inference engine is useful for designing different expert systems.

AI Lec05/3

3



Working memory

- The working memory is also called ***short-term memory.***
- It contains the observed data for making decisions, and the intermediate results (derived data) ***produced*** by the inference engine.

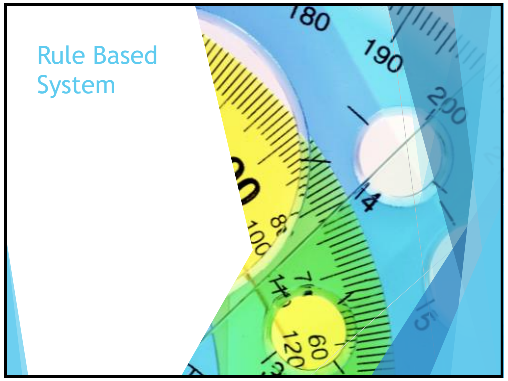
Example:

- f1: x has black strings
- f2: x has white body color
- f3: x has hoofs
- f4: x has odd number of toes
- f5: x drink milk

A child is often considered as "clever" if he/she has a good short memory capability.

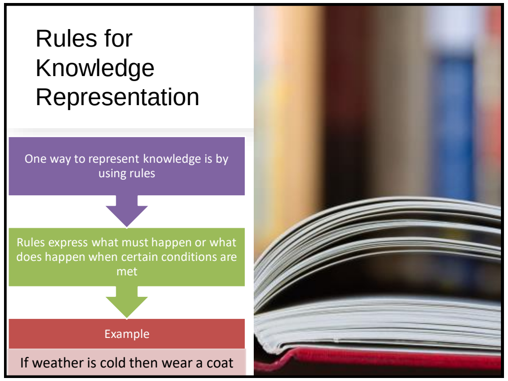
AI Lec05/4

4



Rule Based System

5



Rules for Knowledge Representation

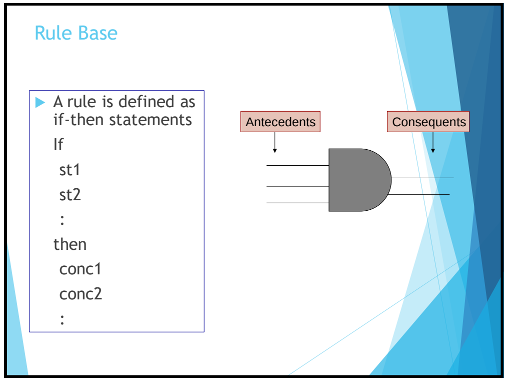
One way to represent knowledge is by using rules

Rules express what must happen or what does happen when certain conditions are met

Example

If weather is cold then wear a coat

6



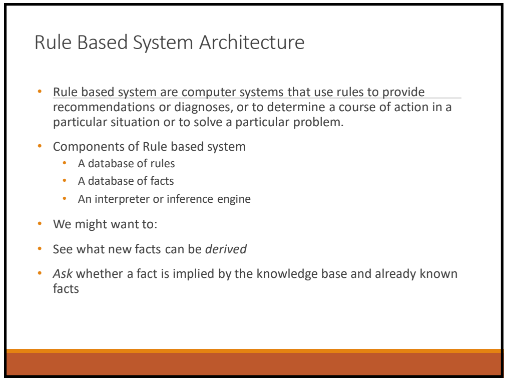
Rule Base

- A rule is defined as if-then statements

If
st1
st2
:
then
conc1
conc2
:

Antecedents
Consequents

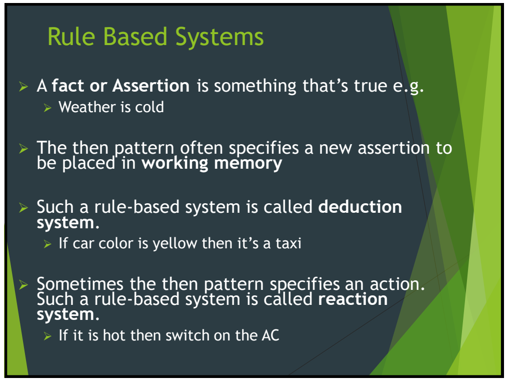
7



Rule Based System Architecture

- Rule based system are computer systems that use rules to provide recommendations or diagnoses, or to determine a course of action in a particular situation or to solve a particular problem.
- Components of Rule based system
  - A database of rules
  - A database of facts
  - An interpreter or inference engine
- We might want to:
- See what new facts can be *derived*
- *Ask* whether a fact is implied by the knowledge base and already known facts

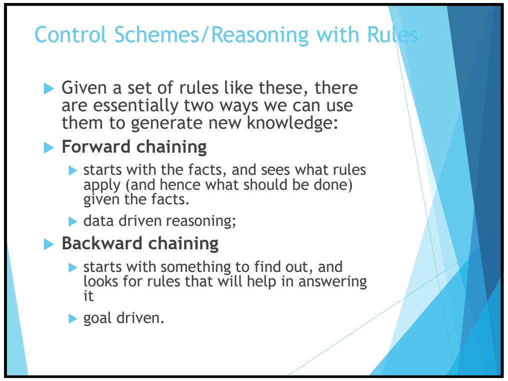
8



Rule Based Systems

- A **fact or Assertion** is something that's true e.g.
  - Weather is cold
- The then pattern often specifies a new assertion to be placed in **working memory**
- Such a rule-based system is called **deduction system.**
  - If car color is yellow then it's a taxi
- Sometimes the then pattern specifies an action. Such a rule-based system is called **reaction system.**
  - If it is hot then switch on the AC

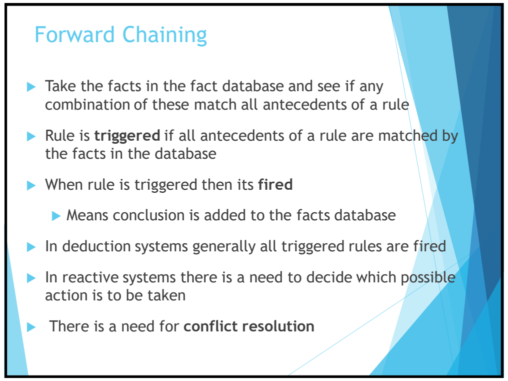
9



Control Schemes/Reasoning with Rules

- Given a set of rules like these, there are essentially two ways we can use them to generate new knowledge:
- **Forward chaining**
  - starts with the facts, and sees what rules apply (and hence what should be done) given the facts.
  - data driven reasoning;
- **Backward chaining**
  - starts with something to find out, and looks for rules that will help in answering it
  - goal driven.

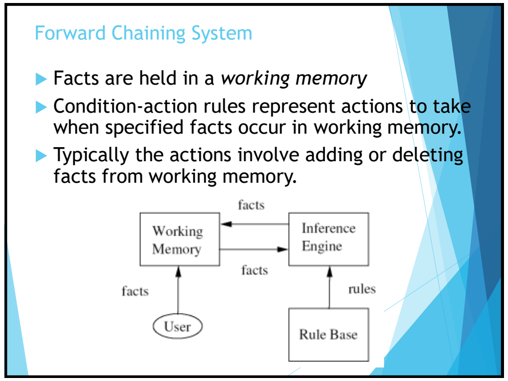
10



Forward Chaining

- Take the facts in the fact database and see if any combination of these match all antecedents of a rule
- Rule is **triggered** if all antecedents of a rule are matched by the facts in the database
- When rule is triggered then its **fired**
  - Means conclusion is added to the facts database
- In deduction systems generally all triggered rules are fired
- In reactive systems there is a need to decide which possible action is to be taken
- There is a need for **conflict resolution**

11

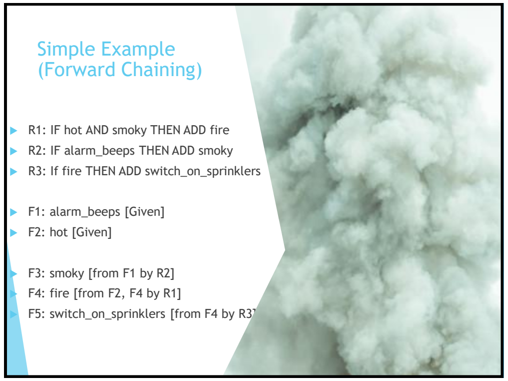


Forward Chaining System

- Facts are held in a *working memory*
- Condition-action rules represent actions to take when specified facts occur in working memory.
- Typically the actions involve adding or deleting facts from working memory.

facts
Working Memory
Inference Engine
facts
facts
rules
User
Rule Base

12



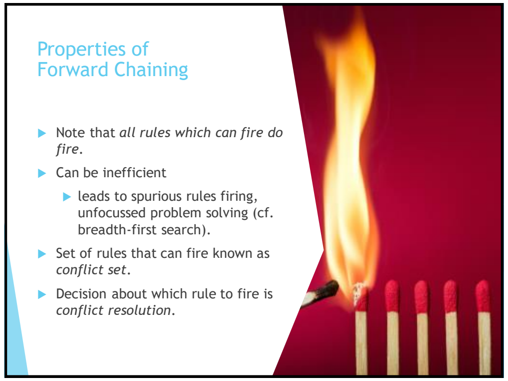
Simple Example (Forward Chaining)

- R1: IF hot AND smoky THEN ADD fire
- R2: IF alarm_beeps THEN ADD smoky
- R3: If fire THEN ADD switch_on_sprinklers

- F1: alarm_beeps [Given]
- F2: hot [Given]

- F3: smoky [from F1 by R2]
- F4: fire [from F2, F4 by R1]
- F5: switch_on_sprinklers [from F4 by R3]

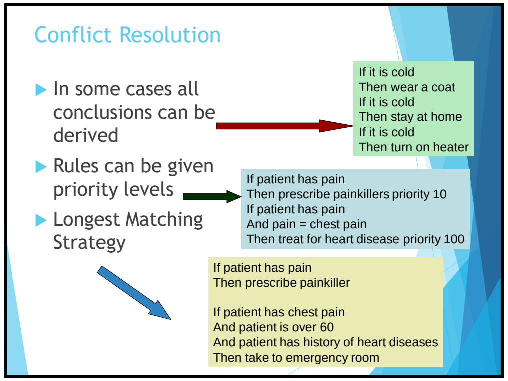
13



Properties of Forward Chaining

- Note that *all rules which can fire do fire*.
- Can be inefficient
  - leads to spurious rules firing, unfocussed problem solving (cf. breadth-first search).
- Set of rules that can fire known as *conflict set*.
- Decision about which rule to fire is *conflict resolution*.

14



Conflict Resolution

- In some cases all conclusions can be derived
- Rules can be given priority levels
- Longest Matching Strategy

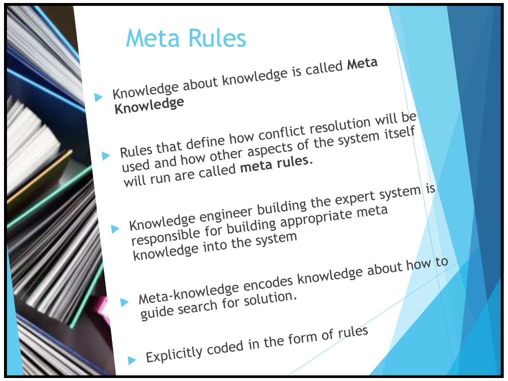
If it is cold
Then wear a coat
If it is cold
Then stay at home
If it is cold
Then turn on heater

If patient has pain
Then prescribe painkillers priority 10
If patient has pain
And pain = chest pain
Then treat for heart disease priority 100

If patient has pain
Then prescribe painkiller

If patient has chest pain
And patient is over 60
And patient has history of heart diseases
Then take to emergency room

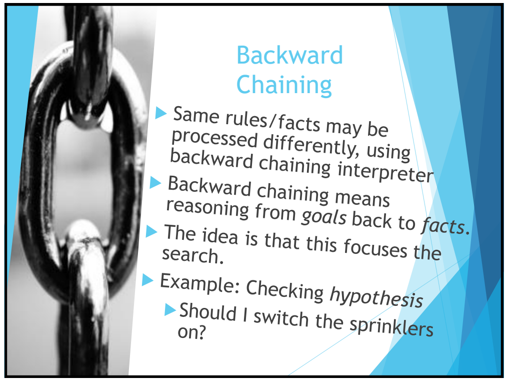
15



Meta Rules

- Knowledge about knowledge is called **Meta Knowledge**
- Rules that define how conflict resolution will be used and how other aspects of the system itself will run are called **meta rules**.
- Knowledge engineer building the expert system is responsible for building appropriate meta knowledge into the system
- Meta-knowledge encodes knowledge about how to guide search for solution.
- Explicitly coded in the form of rules

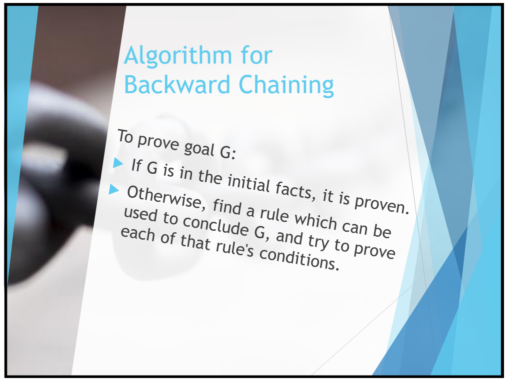
16



Backward Chaining

- Same rules/facts may be processed differently, using backward chaining interpreter
- Backward chaining means reasoning from *goals* back to *facts*.
- The idea is that this focuses the search.
- Example: Checking *hypothesis*
  - Should I switch the sprinklers on?

17

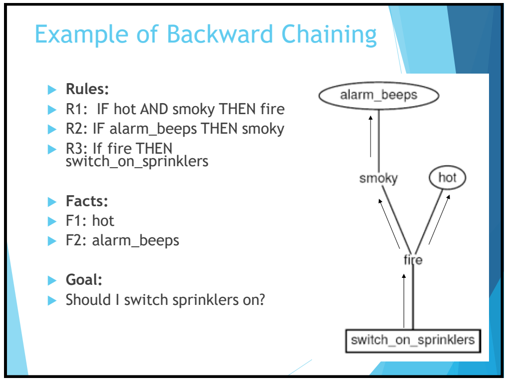


Algorithm for Backward Chaining

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.

18



Example of Backward Chaining

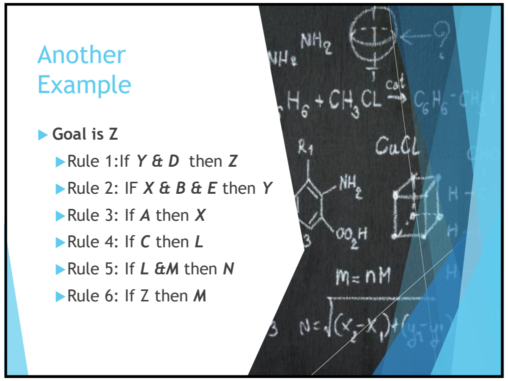
- **Rules:**
- R1: IF hot AND smoky THEN fire
- R2: IF alarm_beeps THEN smoky
- R3: If fire THEN switch_on_sprinklers

- **Facts:**
- F1: hot
- F2: alarm_beeps

- **Goal:**
- Should I switch sprinklers on?

alarm_beeps
smoky
hot
fire
switch_on_sprinklers

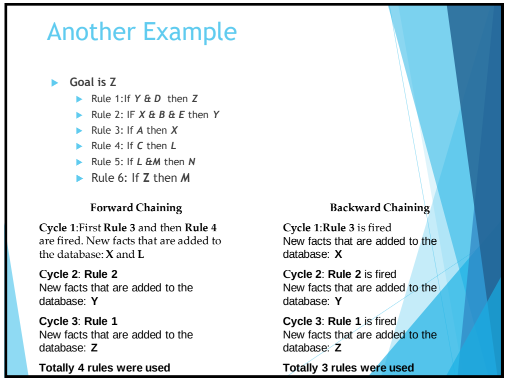
19



Another Example

- **Goal is Z**
  - Rule 1:If ***Y & D*** then ***Z***
  - Rule 2: IF ***X & B & E*** then ***Y***
  - Rule 3: If ***A*** then ***X***
  - Rule 4: If ***C*** then ***L***
  - Rule 5: If ***L &M*** then ***N***
  - Rule 6: If Z then ***M***

20



Another Example

- Goal is Z
  - Rule 1:If ***Y & D*** then ***Z***
  - Rule 2: IF ***X & B & E*** then ***Y***
  - Rule 3: If ***A*** then ***X***
  - Rule 4: If ***C*** then ***L***
  - Rule 5: If ***L &M*** then ***N***
  - Rule 6: If Z then ***M***

Forward Chaining

Cycle 1:First **Rule 3** and then **Rule 4** are fired. New facts that are added to the database: **X** and **L**

Cycle 2: **Rule 2**
New facts that are added to the database: **Y**

Cycle 3: **Rule 1**
New facts that are added to the database: **Z**

Totally 4 rules were used

Backward Chaining

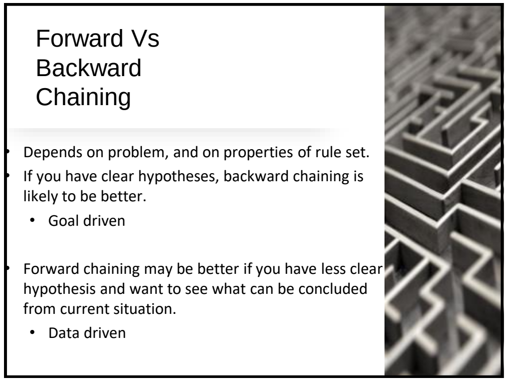
Cycle 1:**Rule 3** is fired
New facts that are added to the database: **X**

Cycle 2: **Rule 2** is fired
New facts that are added to the database: **Y**

Cycle 3: **Rule 1** is fired
New facts that are added to the database: **Z**

Totally 3 rules were used

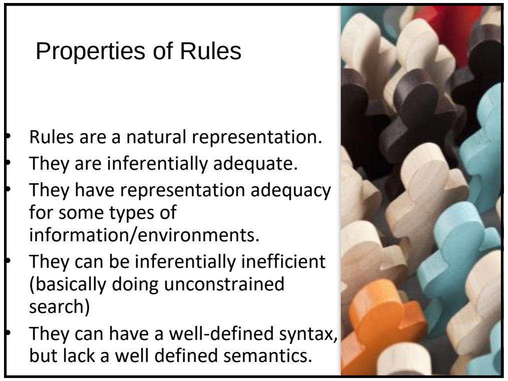
21



Forward Vs Backward Chaining

- Depends on problem, and on properties of rule set.
- If you have clear hypotheses, backward chaining is likely to be better.
  - Goal driven
- Forward chaining may be better if you have less clear hypothesis and want to see what can be concluded from current situation.
  - Data driven

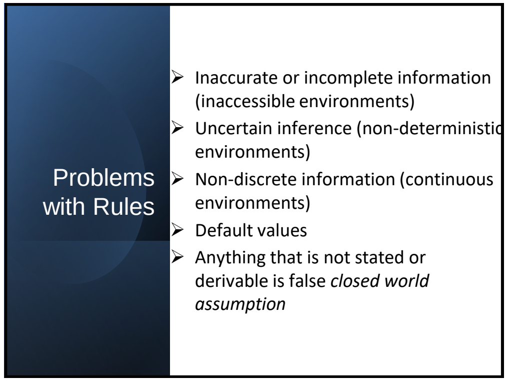
22



Properties of Rules

- Rules are a natural representation.
- They are inferentially adequate.
- They have representation adequacy for some types of information/environments.
- They can be inferentially inefficient (basically doing unconstrained search)
- They can have a well-defined syntax, but lack a well defined semantics.

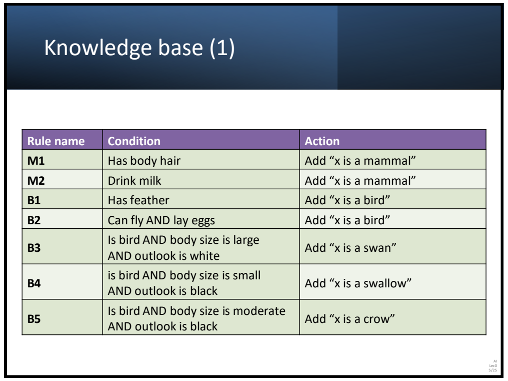
23



Problems with Rules

- Inaccurate or incomplete information (inaccessible environments)
- Uncertain inference (non-deterministic environments)
- Non-discrete information (continuous environments)
- Default values
- Anything that is not stated or derivable is false *closed world assumption*

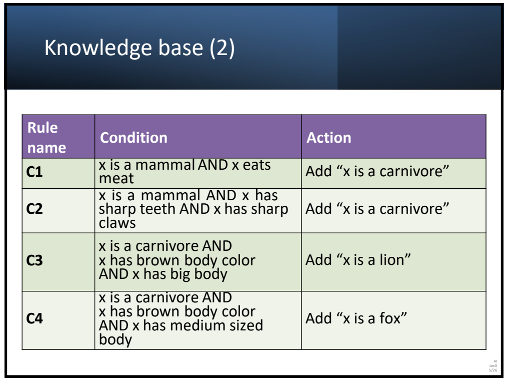
24



Knowledge base (1)

Rule name	Condition	Action
M1	Has body hair	Add “x is a mammal”
M2	Drink milk	Add “x is a mammal”
B1	Has feather	Add “x is a bird”
B2	Can fly AND lay eggs	Add “x is a bird”
B3	Is bird AND body size is large AND outlook is white	Add “x is a swan”
B4	is bird AND body size is small AND outlook is black	Add “x is a swallow”
B5	Is bird AND body size is moderate AND outlook is black	Add “x is a crow”

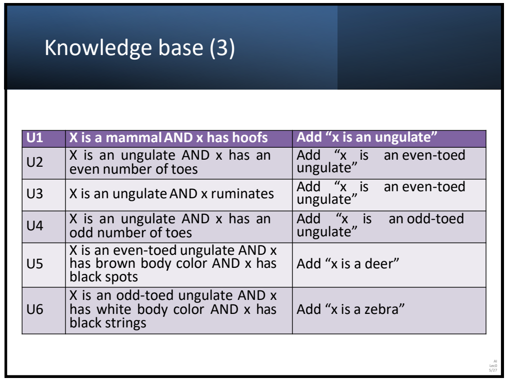
25



Knowledge base (2)

Rule name	Condition	Action
C1	x is a mammal AND x eats meat	Add “x is a carnivore”
C2	x is a mammal AND x has sharp teeth AND x has sharp claws	Add “x is a carnivore”
C3	x is a carnivore AND x has brown body color AND x has big body	Add “x is a lion”
C4	x is a carnivore AND x has brown body color AND x has medium sized body	Add “x is a fox”

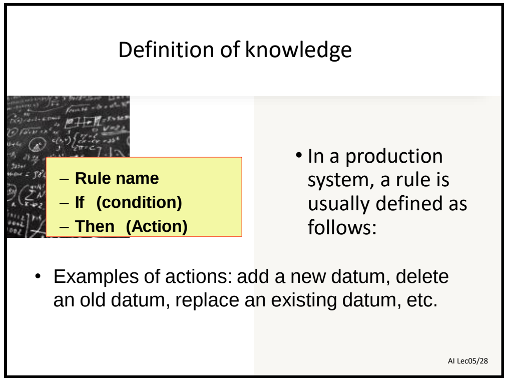
26



Knowledge base (3)

U1	X is a mammal AND x has hoofs	Add “x is an ungulate”
U2	X is an ungulate AND x has an even number of toes	Add “x is an even-toed ungulate”
U3	X is an ungulate AND x ruminates	Add “x is an even-toed ungulate”
U4	X is an ungulate AND x has an odd number of toes	Add “x is an odd-toed ungulate”
U5	X is an even-toed ungulate AND x has brown body color AND x has black spots	Add “x is a deer”
U6	X is an odd-toed ungulate AND x has white body color AND x has black strings	Add “x is a zebra”

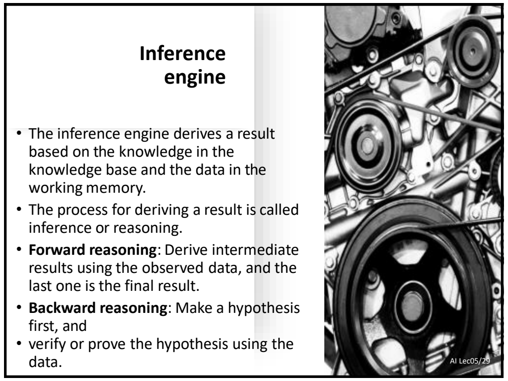
27



Definition of knowledge

- In a production system, a rule is usually defined as follows:
  - Rule name
  - If (condition)
  - Then (Action)
- Examples of actions: add a new datum, delete an old datum, replace an existing datum, etc.

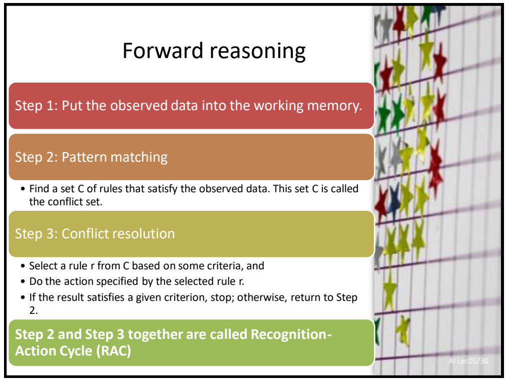
28



Inference engine

- The inference engine derives a result based on the knowledge in the knowledge base and the data in the working memory.
- The process for deriving a result is called inference or reasoning.
- **Forward reasoning**: Derive intermediate results using the observed data, and the last one is the final result.
- **Backward reasoning**: Make a hypothesis first, and
- verify or prove the hypothesis using the data.

29



Forward reasoning

Step 1: Put the observed data into the working memory.

Step 2: Pattern matching

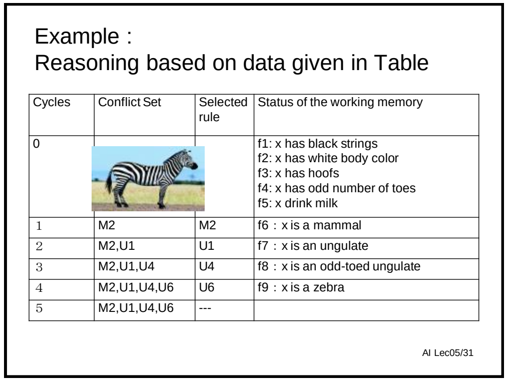
- Find a set C of rules that satisfy the observed data. This set C is called the conflict set.

Step 3: Conflict resolution

- Select a rule r from C based on some criteria, and
- Do the action specified by the selected rule r.
- If the result satisfies a given criterion, stop; otherwise, return to Step 2.

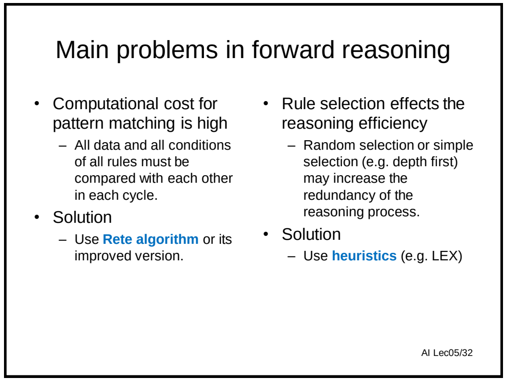
Step 2 and Step 3 together are called Recognition-Action Cycle (RAC)

30



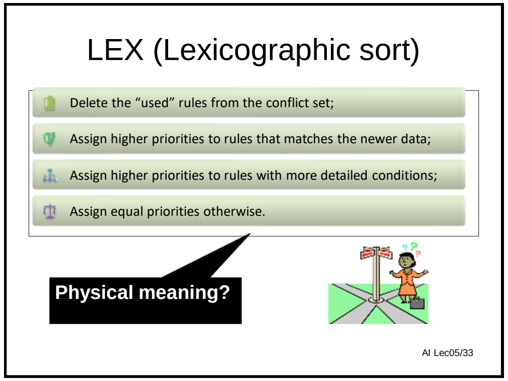
Example : Reasoning based on data given in Table

Cycles	Conflict Set	Selected rule	Status of the working memory
0			f1: x has black strings f2: x has white body color f3: x has hoofs f4: x has odd number of toes f5: x drink milk
1	M2	M2	f6：x is a mammal
2	M2,U1	U1	f7：x is an ungulate
3	M2,U1,U4	U4	f8：x is an odd-toed ungulate
4	M2,U1,U4,U6	U6	f9：x is a zebra
5	M2,U1,U4,U6	---	



Main problems in forward reasoning

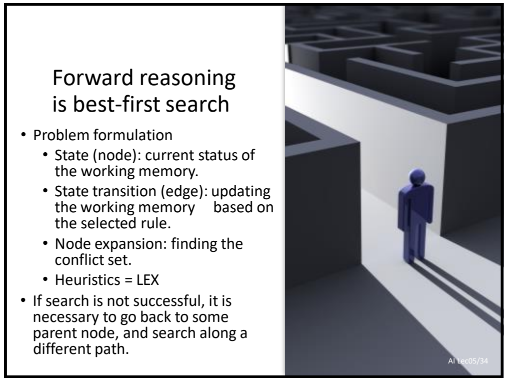
- Computational cost for pattern matching is high
  - All data and all conditions of all rules must be compared with each other in each cycle.
- Solution
  - Use **Rete algorithm** or its improved version.
- Rule selection effects the reasoning efficiency
  - Random selection or simple selection (e.g. depth first) may increase the redundancy of the reasoning process.
- Solution
  - Use **heuristics** (e.g. LEX)



LEX (Lexicographic sort)

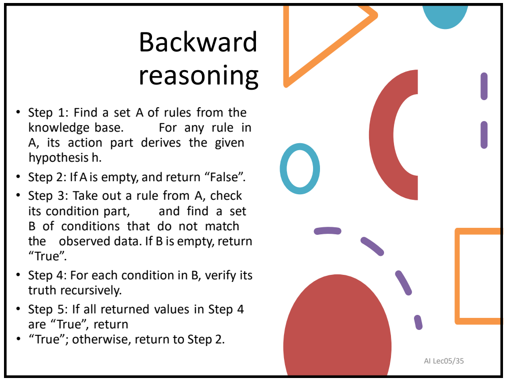
- Delete the “used” rules from the conflict set;
- Assign higher priorities to rules that matches the newer data;
- Assign higher priorities to rules with more detailed conditions;
- Assign equal priorities otherwise.

Physical meaning?



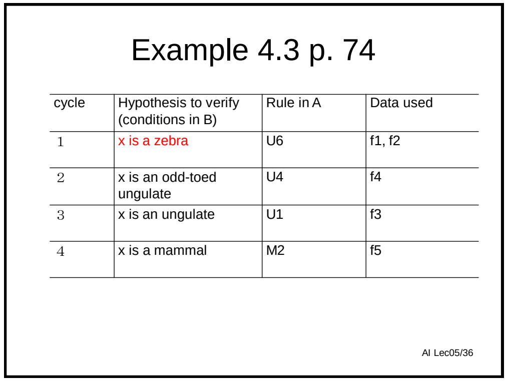
Forward reasoning is best-first search

- Problem formulation
  - State (node): current status of the working memory.
  - State transition (edge): updating the working memory based on the selected rule.
  - Node expansion: finding the conflict set.
  - Heuristics = LEX
- If search is not successful, it is necessary to go back to some parent node, and search along a different path.



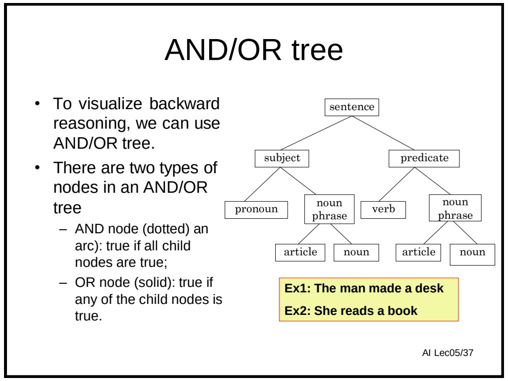
Backward reasoning

- Step 1: Find a set A of rules from the knowledge base. For any rule in A, its action part derives the given hypothesis h.
- Step 2: If A is empty, and return “False”.
- Step 3: Take out a rule from A, check its condition part, and find a set B of conditions that do not match the observed data. If B is empty, return “True”.
- Step 4: For each condition in B, verify its truth recursively.
- Step 5: If all returned values in Step 4 are “True”, return
- “True”; otherwise, return to Step 2.



Example 4.3 p. 74

cycle	Hypothesis to verify (conditions in B)	Rule in A	Data used
1	x is a zebra	U6	f1, f2
2	x is an odd-toed ungulate	U4	f4
3	x is an ungulate	U1	f3
4	x is a mammal	M2	f5



AND/OR tree

- To visualize backward reasoning, we can use AND/OR tree.
- There are two types of nodes in an AND/OR tree
  - AND node (dotted) an arc): true if all child nodes are true;
  - OR node (solid): true if any of the child nodes is true.

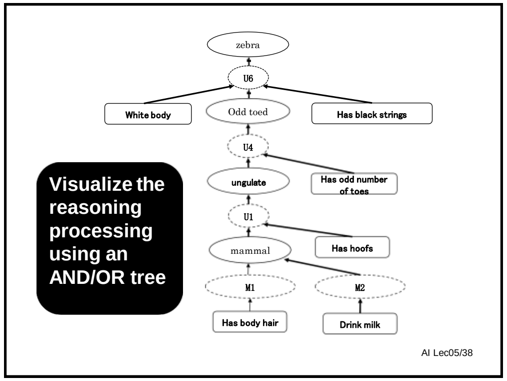
sentence
subject
predicate
pronoun
noun phrase
verb
noun phrase
article
noun
article
noun

Ex1: The man made a desk

Ex2: She reads a book

AI Lec05/37

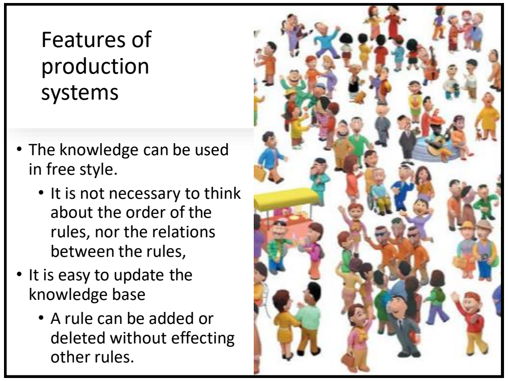
37



Visualize the reasoning processing using an AND/OR tree

AI Lec05/38

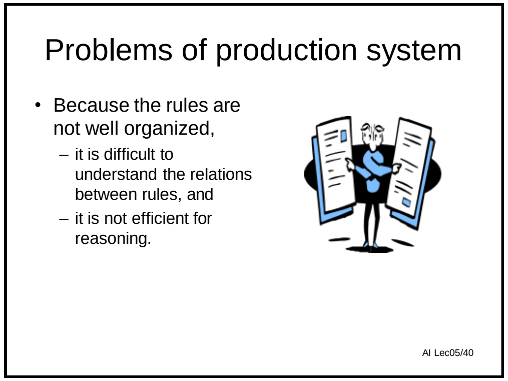
38



Features of production systems

- The knowledge can be used in free style.
  - It is not necessary to think about the order of the rules, nor the relations between the rules,
- It is easy to update the knowledge base
  - A rule can be added or deleted without effecting other rules.

39

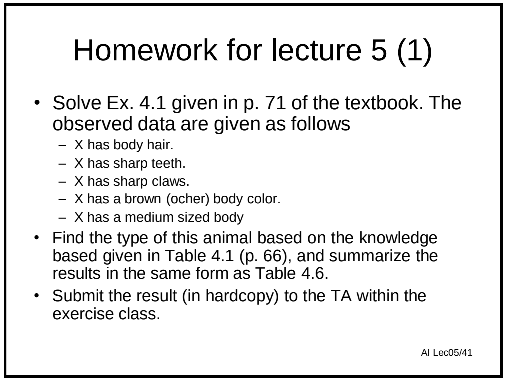


Problems of production system

- Because the rules are not well organized,
  - it is difficult to understand the relations between rules, and
  - it is not efficient for reasoning.

AI Lec05/40

40

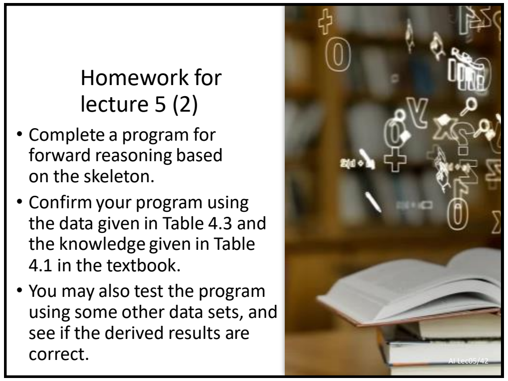


Homework for lecture 5 (1)

- Solve Ex. 4.1 given in p. 71 of the textbook. The observed data are given as follows
  - X has body hair.
  - X has sharp teeth.
  - X has sharp claws.
  - X has a brown (ocher) body color.
  - X has a medium sized body
- Find the type of this animal based on the knowledge based given in Table 4.1 (p. 66), and summarize the results in the same form as Table 4.6.
- Submit the result (in hardcopy) to the TA within the exercise class.

AI Lec05/41

41

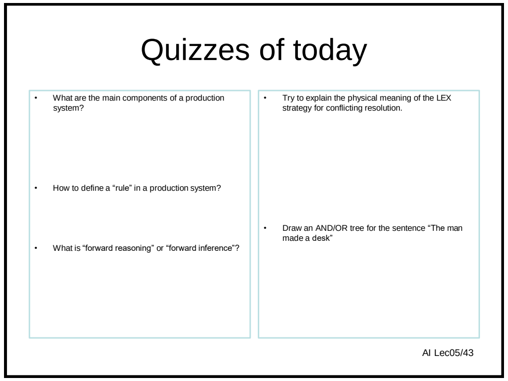


Homework for lecture 5 (2)

- Complete a program for forward reasoning based on the skeleton.
- Confirm your program using the data given in Table 4.3 and the knowledge given in Table 4.1 in the textbook.
- You may also test the program using some other data sets, and see if the derived results are correct.

AI Lec05/42

42



Quizzes of today

- What are the main components of a production system?
- How to define a “rule” in a production system?
- What is “forward reasoning” or “forward inference”?

- Try to explain the physical meaning of the LEX strategy for conflicting resolution.
- Draw an AND/OR tree for the sentence “The man made a desk”

AI Lec05/43

43