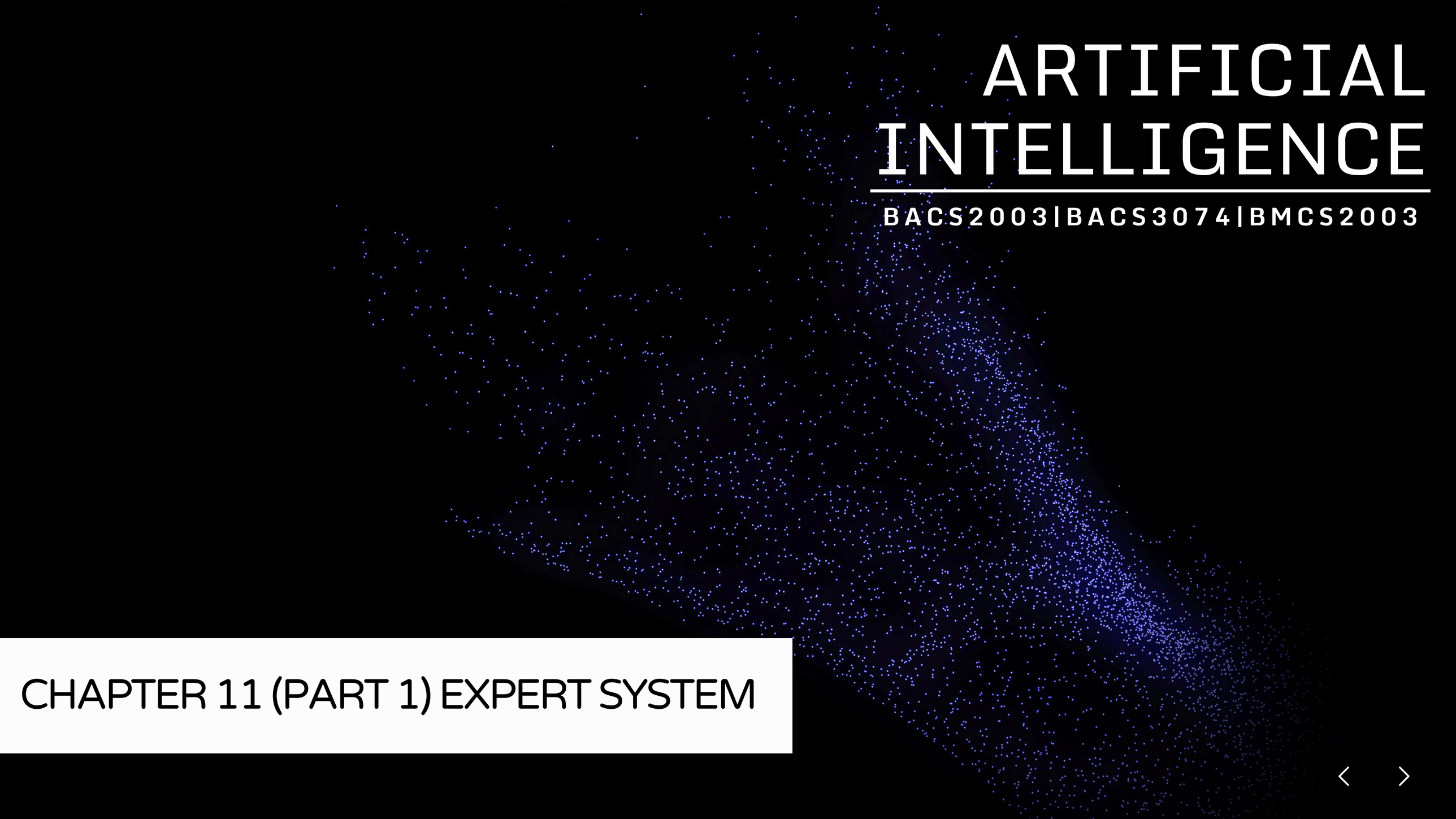
ARTIFICIAL INTELLIGENCE

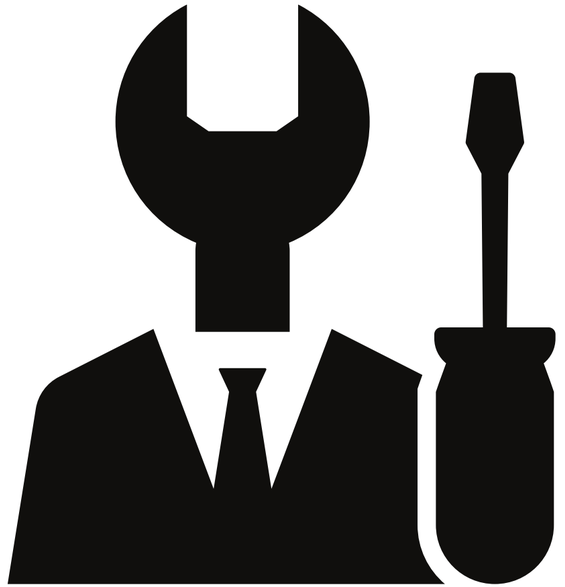
BACS2003|BACS3074|BMCS2003



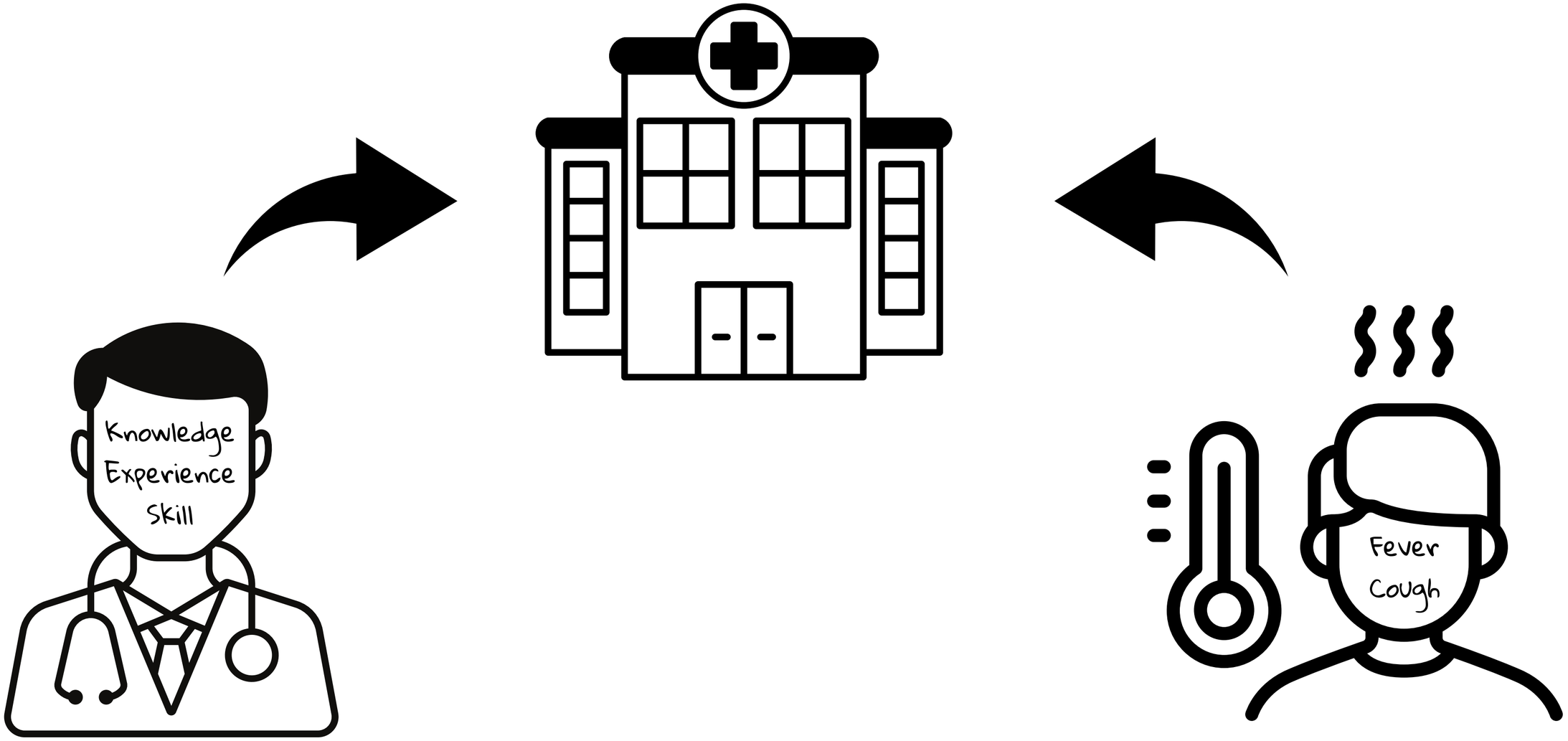
CHAPTER 11 (PART 1) EXPERT SYSTEM

OUTCOMES

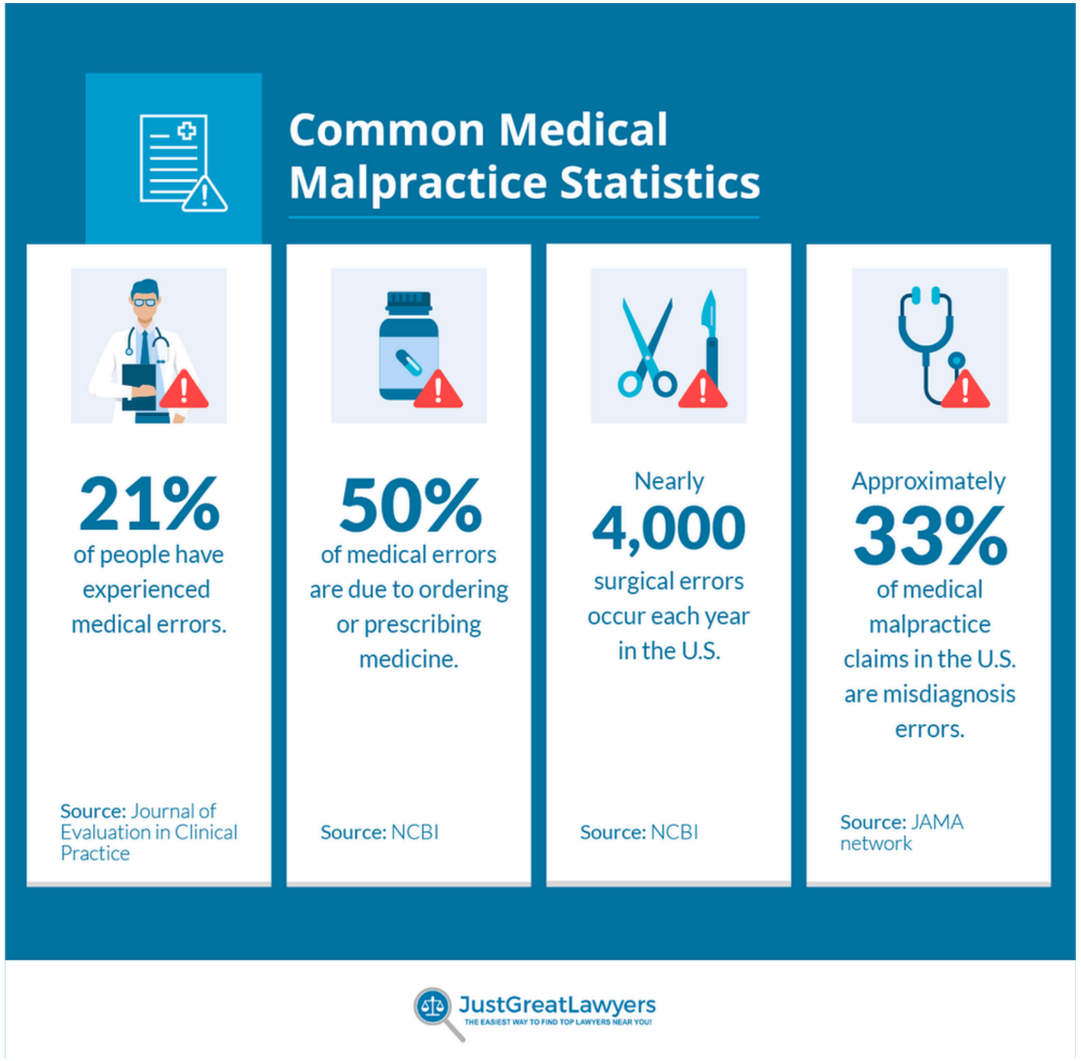
1. **Introduction of Expert System**
2. **Expert System Architecture**
3. **Conflict Resolution & Reasoning Mechanisms**



HUMAN EXPERT



HUMAN EXPERT



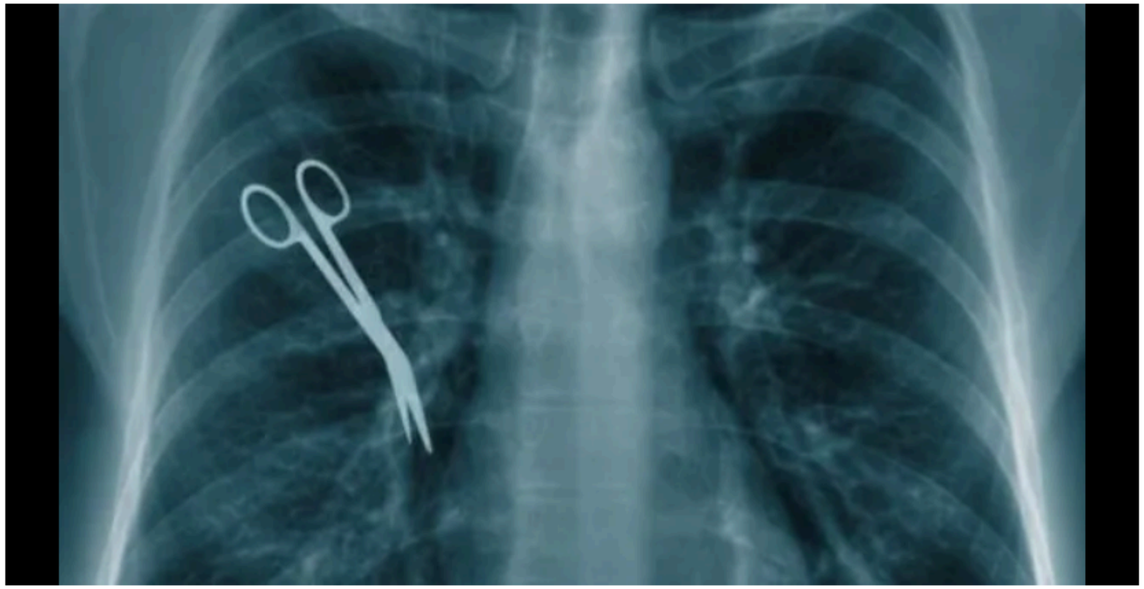
NEWS

You won't believe these Indiana medical errors

Shari Rudavsky shari.rudavsky@indystar.com

Published 11:41 a.m. ET Oct. 7, 2014 | **Updated 1:19 p.m. ET Oct. 7, 2014**

View Comments



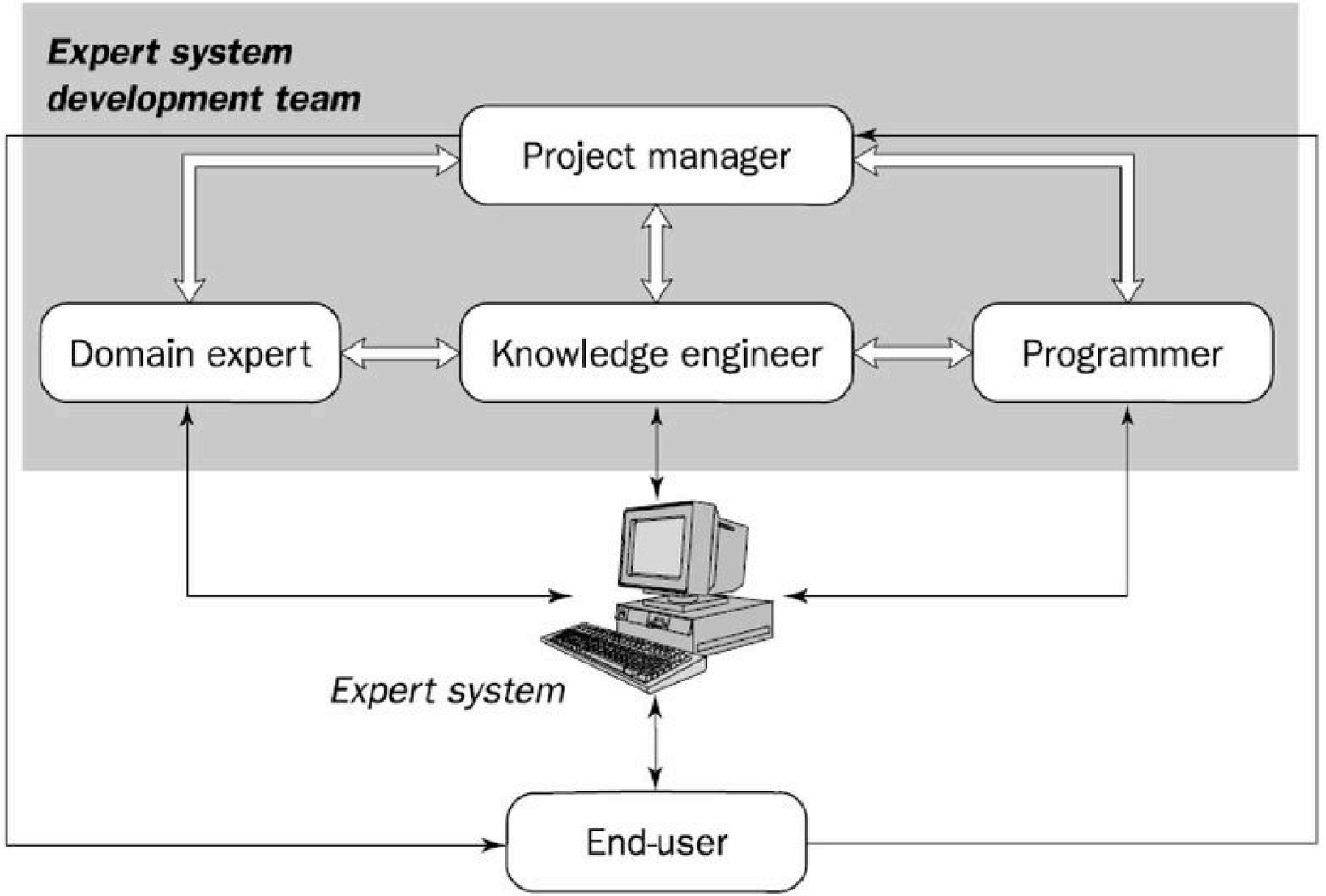
This chest X-ray shows a pair of surgical scissors left behind in a patient. Indiana hospital errors in 2014 were the highest in nine years of reporting. *Lukasz Panek / Getty Images / IStockphoto*

EXPERT SYSTEM VS HUMAN EXPERT

Aspect	Expert System	Human Expert
Knowledge Processing	Processes knowledge expressed in the form of rules and uses symbolic reasoning in a narrow domain	Uses knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain
Explanation	Helps in tracing the rules that were produced during problem-solving and can explain the reasoning	Capable of explaining the line of reasoning and providing the details
Reasoning with Uncertain Information	Permits inexact reasoning and can deal with incomplete, uncertain, and fuzzy data	Uses inexact reasoning and is also able to deal with incomplete, uncertain, and fuzzy information
Availability	Always accessible	Available only during certain hours of the day
Speed	Handles any problem in a concise amount of time	Can take their time to solve problems
Consistency	Consistent	Can be unpredictable
Documentation	Easy to document	Can be difficult to document
Transferability	Knowledge is transferable	Knowledge transfer can be difficult

While expert systems have many advantages, they are not intended to replace human experts but to complement them. Each has its strengths and limitations, and the choice between using an expert system or a human expert depends on the specific context and requirements

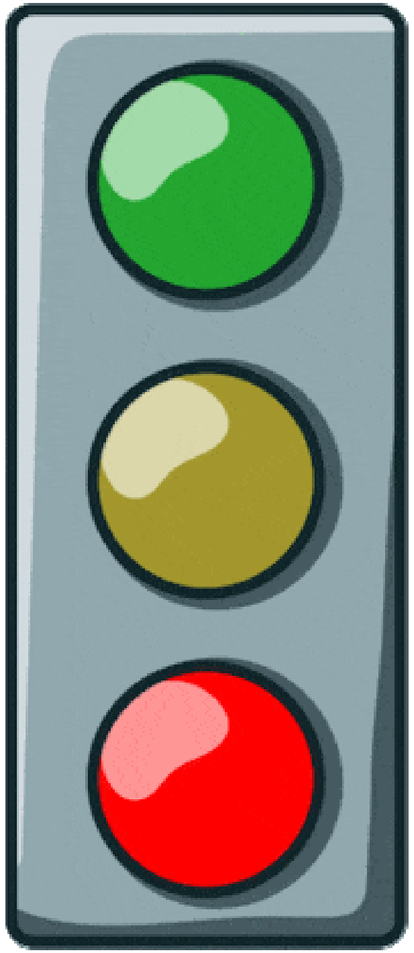
MAIN PLAYER IN EXPERT SYSTEM DEVELOPMENT TEAM



RULES AS KNOWLEDGE REPRESENTATION

IF the ‘traffic light’ is green
THEN the action is go

IF the ‘traffic light’ is red
THEN the action is stop



RULES AS KNOWLEDGE REPRESENTATION

Any rule consists of two parts:

- IF part, called the antecedent (premise or condition)
- THEN part called the consequent (conclusion or action).

```
IF      <antecedent>
THEN    <consequent>
```

USING CONJUNCTION AND DISJUNCTION

A rule can have multiple antecedents joined by the keywords

- AND (conjunction)
- OR (disjunction)
- a combination of both.

```
IF      <antecedent 1>
AND     <antecedent 2>
          ...
AND     <antecedent>
THEN    <consequent>
```

```
IF      <antecedent 1>
OR      <antecedent 2>
          ...
OR      <antecedent>
THEN    <consequent>
```

TYPES OF RULES

- Relations
- Recommendations
- Heuristics
- Directive
- Strategies

TYPES OF RULES

Relations

IF

the ‘fuel tank’ is empty

THEN

the car is dead

TYPES OF RULES

Recommendations

IF

the season is autumn

AND

the sky is cloudy

AND

the forecast is drizzle

THEN

the advice is ‘take an umbrella’

TYPES OF RULES

Heuristics

IF

the spill is liquid

AND

the ‘spill pH’ < 6

AND

the ‘spill smell’ is vinegar

THEN

the ‘spill material’ is ‘acetic acid’

TYPES OF RULES

Directive

IF

the car is dead

AND

the ‘fuel tank’ is empty

THEN

the action is ‘refuel the car’

TYPES OF RULES

Strategies

IF the car is dead

THEN the action is ‘check the fuel tank’ => step1 is complete

IF step1 is complete

AND the ‘fuel tank’ is full

THEN the action is ‘check the battery’ => step2 is complete

QUESTIONS

relations
recommendations
heuristics
directives
strategies

Identify each type of rules below

1

if cough present is true
and phlegm has blood stain
then symptom of TB is true

2

if diagnose is TB
then prescribe anti-TB drugs

3

if symptom of TB is true
then go for chest X-ray

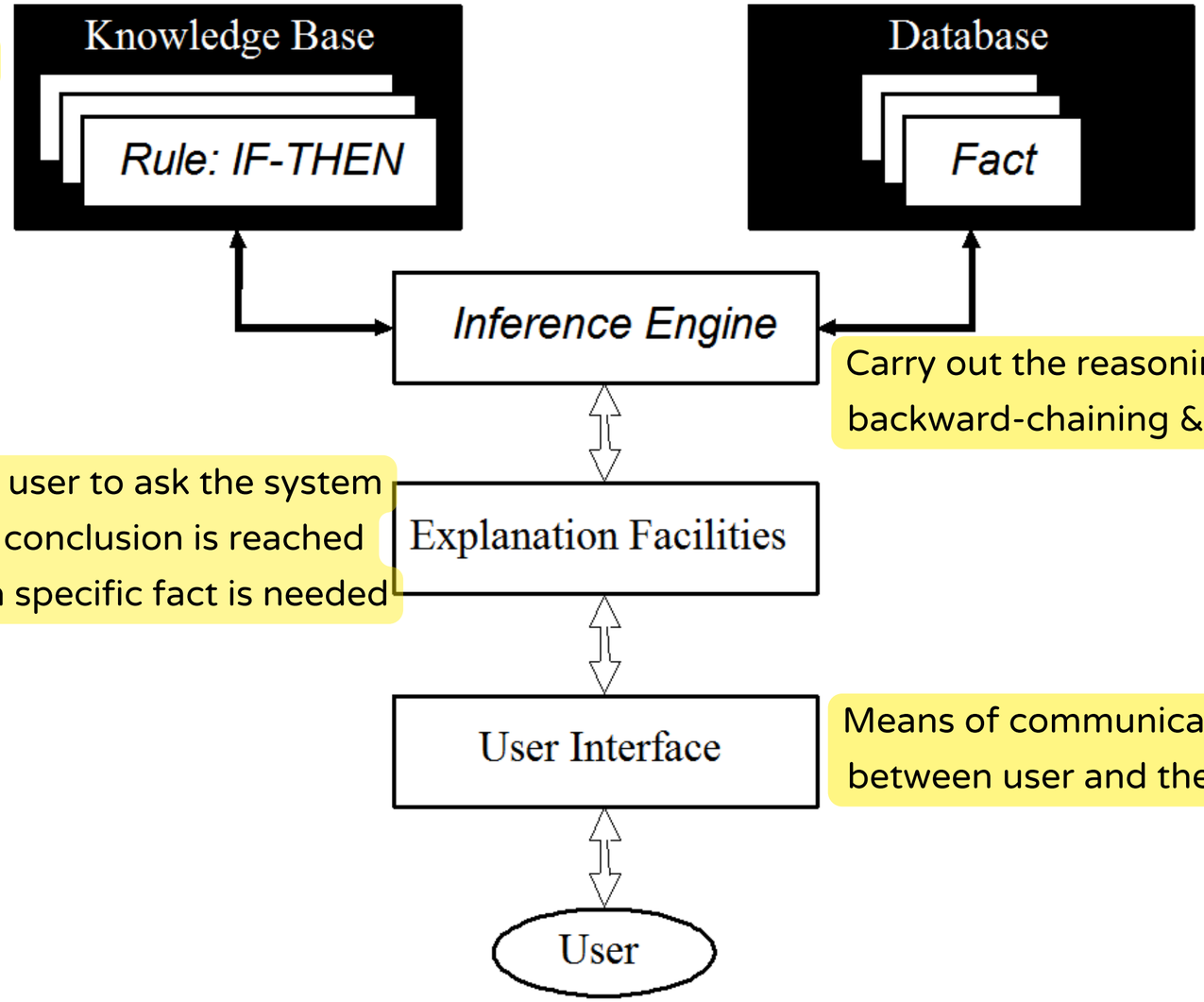
4

if TB is true
then cough present is true

STRUCTURE OF A EXPERT SYSTEM

PRODUCTION SYSTEM MODEL

Contains domain knowledge or set of rules, will fire rules to execute necessary action



The set of facts will match with the rules in the knowledge base

Carry out the reasoning. May use forward-chaining, backward-chaining & conflict resolution procedure.

Allow the user to ask the system how the conclusion is reached and why a specific fact is needed

Means of communication between user and the ES

CHARACTERISTICS OF AN EXPERT SYSTEM

- High-Quality Performance: An expert system is built to perform at a human expert level in a narrow, specialised domain. Thus, the most important characteristic of an expert system is its high-quality performance. No matter how fast the system can solve a problem, the user the user will not be satisfied if the result is wrong.

- Speed: The system's ability to quickly reach a solution is vital, especially in time-sensitive situations such as medical emergencies or critical industrial processes

- Heuristic Reasoning: Expert systems use heuristics, which are rules of thumb or simplified principles, to guide their reasoning process and narrow down the search for solutions

- Explanation Capability: A distinctive feature of expert systems is their ability to explain their reasoning and decisions. This transparency helps users understand and trust the system's output

ADVANTAGES OF EXPERT SYSTEMS

Natural knowledge representation

- An expert usually explains the problem-solving procedure with such expressions as this: “In such-and-such situation, I do so-and-so”. These expressions can be represented quite naturally as IF-THEN production rules.

Uniform structure

- Production rules have the uniform IF-THEN structure. Each rule is an independent piece of knowledge. The very syntax of production rules enables them to be self-documented.

Separation of knowledge from its processing

- The structure of a rule-based expert system provides an effective separation of the knowledge base from the inference engine. This makes it possible to develop different applications using the same expert system shell.

Dealing with incomplete and uncertain knowledge

- Most rule-based expert systems are capable of representing and reasoning with incomplete and uncertain knowledge.

DISADVANTAGES OF EXPERT SYSTEMS

Opaque relations between rules

- Although the individual production rules are relatively simple and self-documented, their logical interactions within the large set of rules may be opaque.
- Rule-based systems make it difficult to observe how individual rules serve the overall strategy.

Ineffective search

- The inference engine applies an exhaustive search through all the production rules during each cycle.
- Expert systems with a large set of rules (over 100 rules) can be slow, and thus large rule-based systems can be unsuitable for real-time applications.

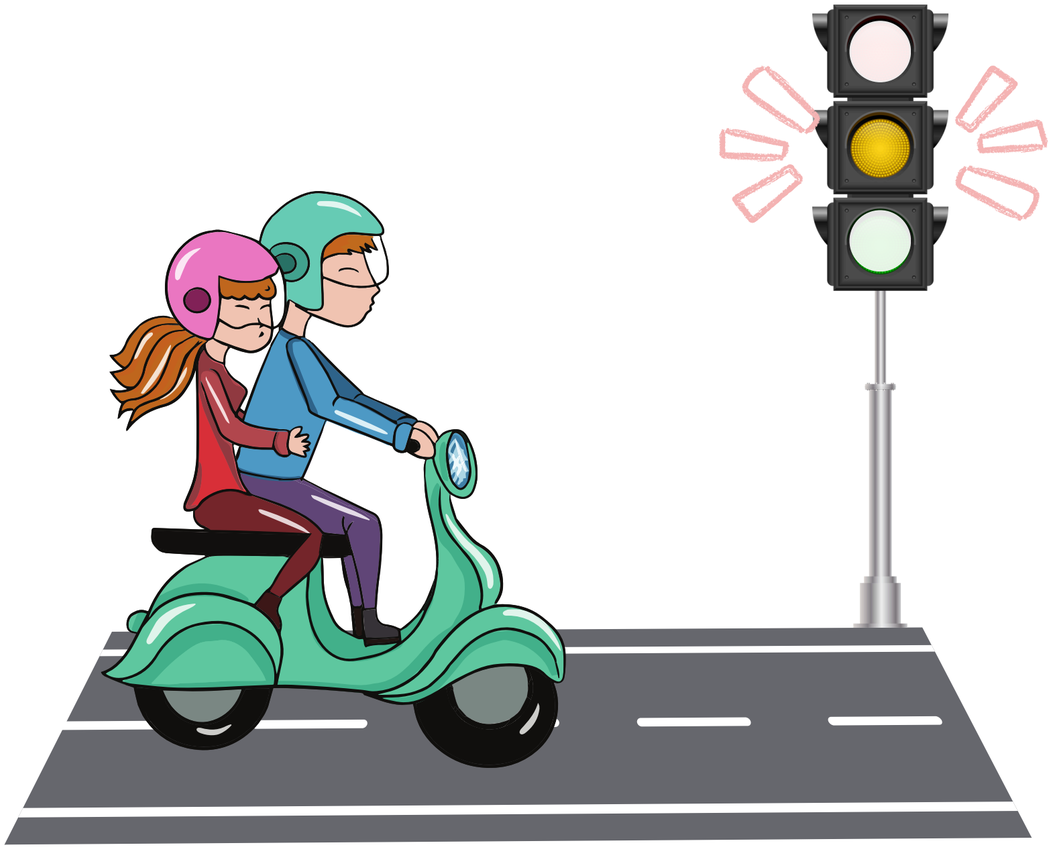
Inability to learn

- In general, rule-based expert systems do not have an ability to learn from the experience.
- Unlike a human expert, who knows when to “break the rules”, an expert system cannot automatically modify its knowledge base, or adjust existing rules or add new ones.
- The knowledge engineer is still responsible for revising and maintaining the system.

CONFLICT RESOLUTION

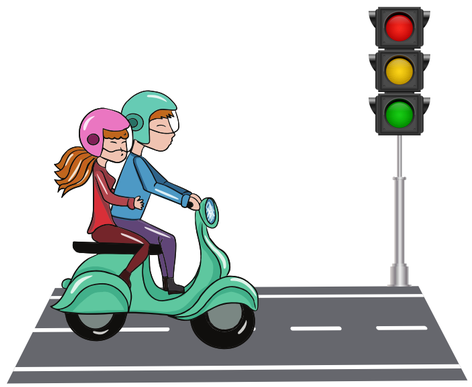
Let’s consider this 3 rules: Which rule to fire?

IF	the ‘traffic light’ is **orange**
THEN	the action is stop
IF	the ‘traffic light’ is **red**
THEN	the action is stop
IF	the ‘traffic light’ is **not red**
THEN	the action is go



CONFLICT RESOLUTION

Let’s consider this 3 rules: Which rule to fire?



IF	the ‘traffic light’ is **orange**
THEN	the action is stop
IF	the ‘traffic light’ is **red**
THEN	the action is stop
IF	the ‘traffic light’ is **not red**
THEN	the action is go

1. If the traffic light is orange:
   - Rule 1 and Rule 3 can be set to fire when the condition part is satisfied.
   - These rules represent a conflict set.

2. The inference engine must determine which rule to fire from such a set.

3. A method for choosing a rule to fire when more than one rule can be fired in a given cycle is called conflict resolution.

CONFLICT RESOLUTION STRATEGIES

1. Highest priority
2. Most specific rules
3. Data most recently entered
4. Meta-rules
5. First come first serve

And many more…

CONFLICT RESOLUTION STRATEGIES

1. Highest priority

- Fire the rule with the highest priority (order of rules).
- Usually this strategy works well for expert systems with around 100 rules.

E.g. for forward chaining inference engine
R1: traffic_light(orange)→stop
R2: traffic_light(red)→stop
R3: traffic_light(~red)→go

R3 has higher priority than R1 (assuming that we set bottom most rule has higher priority)

As a consequence, object action takes new value GO!

CONFLICT RESOLUTION STRATEGIES

2. Most specific rules

- Fire the most specific rule.
- Specific -> longest matching strategy.
- We assume that specific rule possesses more information than a shorter (general) one.

```
IF no_of_conditions(rule A) >
    no_of_conditions(rule B),
THEN rule A is more specific than rule B.
```

R1: has_four_legs(x) ^ mammals(x) ^ bark(x)
-> dog(x)

R2: has_four_legs(x) ^ mammals(x)
-> animal(x)

Facts gathered:

- Pluto has four legs
- Pluto is a mammal
- Pluto can bark

R1 has the most specific rules,
Conclusion: Pluto is a dog

CONFLICT RESOLUTION STRATEGIES

3. Data most recently entered

- Fire the rule that uses the data most recently entered in the database.
- This method relies on time tags attached to each fact in the database.
- In the conflict set, first fires the rule whose antecedent uses the data most recently added to the database.

CONFLICT RESOLUTION STRATEGIES

4. Meta-rules

- Knowledge about knowledge.
- Metaknowledge is knowledge about the use and control of domain knowledge in an expert system.
- In rule-based expert systems, metaknowledge is represented by metarules.

> A metarule determines a strategy for the use of task-specific rules in the expert system.

- Rule 1: If the patient has symptoms of fever and headache, consider the possibility of a flu infection.
- Rule 2: If the patient has symptoms of fever and headache and traveled recently to a tropical area, consider the possibility of malaria.

In this case, a meta rule might be used to resolve a potential conflict between these two rules:
Meta Rule: If the patient has traveled to a tropical area recently, give higher priority to Rule 2 over Rule 1.

This meta rule ensures that the system takes into account the travel history (a critical contextual factor) before considering the more general symptoms of fever and headache, which could apply to many conditions.

CONFLICT RESOLUTION STRATEGIES

5. First come first serve

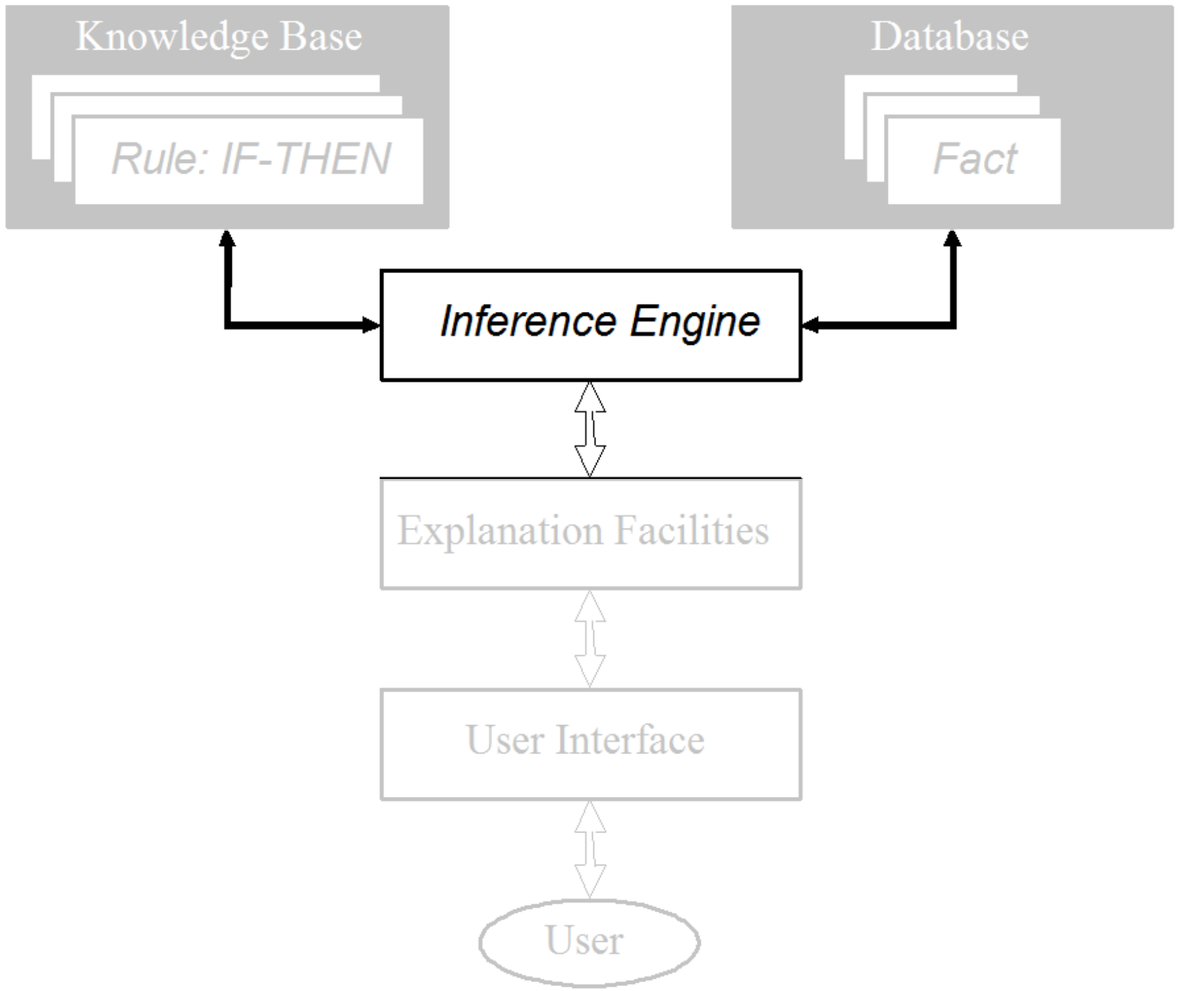
- The first rule that matches the content of the working memory or the first rule that was triggered is the one that gets fired
- This strategy operates based on the order in which the rules were triggered or added to the conflict set. The rule that was triggered first or added first to the conflict set is the one that gets fired first
- However, it's important to note that while the FCFS strategy is simple and easy to implement, it may not always provide the most efficient or fair resolution.

E.g. for forward chaining inference engine
R1: traffic_light(orange)→stop
R2: traffic_light(red)→stop
R3: traffic_light(~red)→go

R1 will be fired first, as a consequence, object action takes new value STOP!

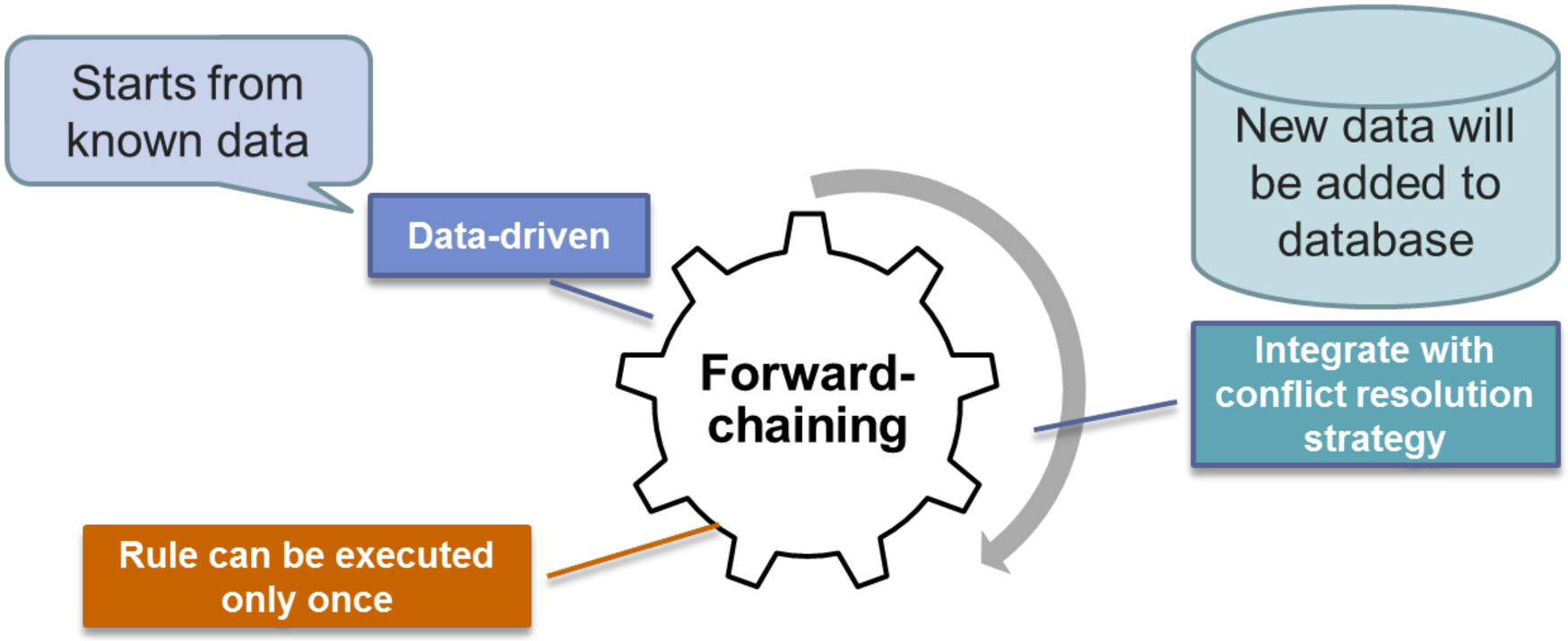
INFERENCE MECHANISM

- Forward-chaining
- Backward-chaining



INFERENCE MECHANISM

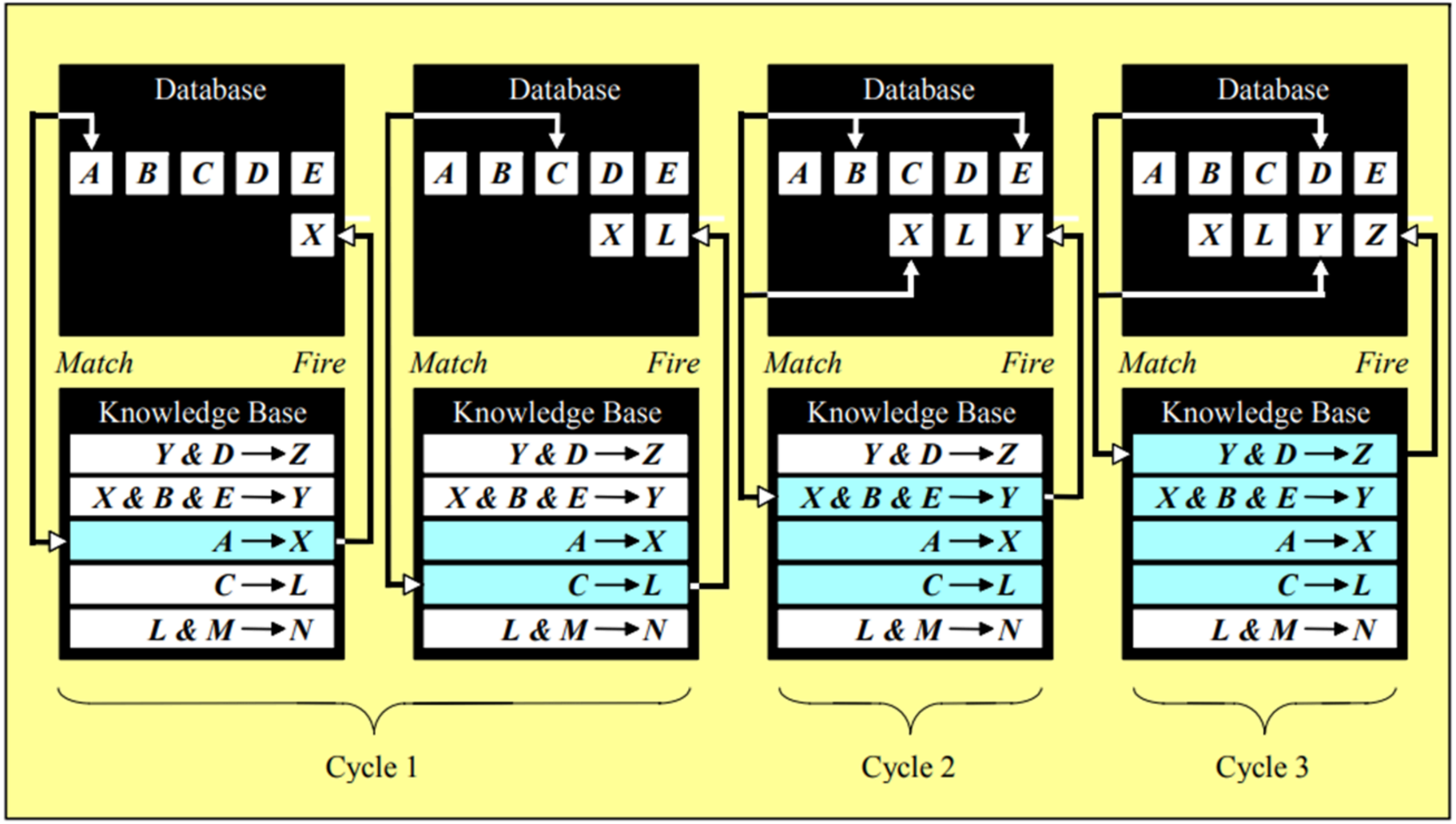
FORWARD-CHAINING



1. Forward chaining is the data-driven reasoning.
2. The reasoning starts from the known data and proceeds forward with that data.
3. Each time only the topmost rule is executed if FCFS strategy is used.
4. When fired, the rule adds a new fact in the database.
5. Any rule can be executed only once.
6. The match-fire cycle stops when no further rules can be fired.

INFERENCE MECHANISM

FORWARD-CHAINING



1. Forward chaining is the data-driven reasoning.
2. The reasoning starts from the known data and proceeds forward with that data.
3. Each time only the topmost rule is executed if highest priority rule is used.
4. When fired, the rule adds a new fact in the database.
5. Any rule can be executed only once.
6. The match-fire cycle stops when no further rules can be fired.

EXAMPLE

FORWARD-CHAINING

Observed facts:

1. Temperature of 102
2. Been sick for 2 months
3. Has a sore throat

EXAMPLE

FORWARD-CHAINING

- *Rule 1*

  IF the patient has a sore throat
  AND suspected a bacterial infection,
  THEN the patient has a strep throat.

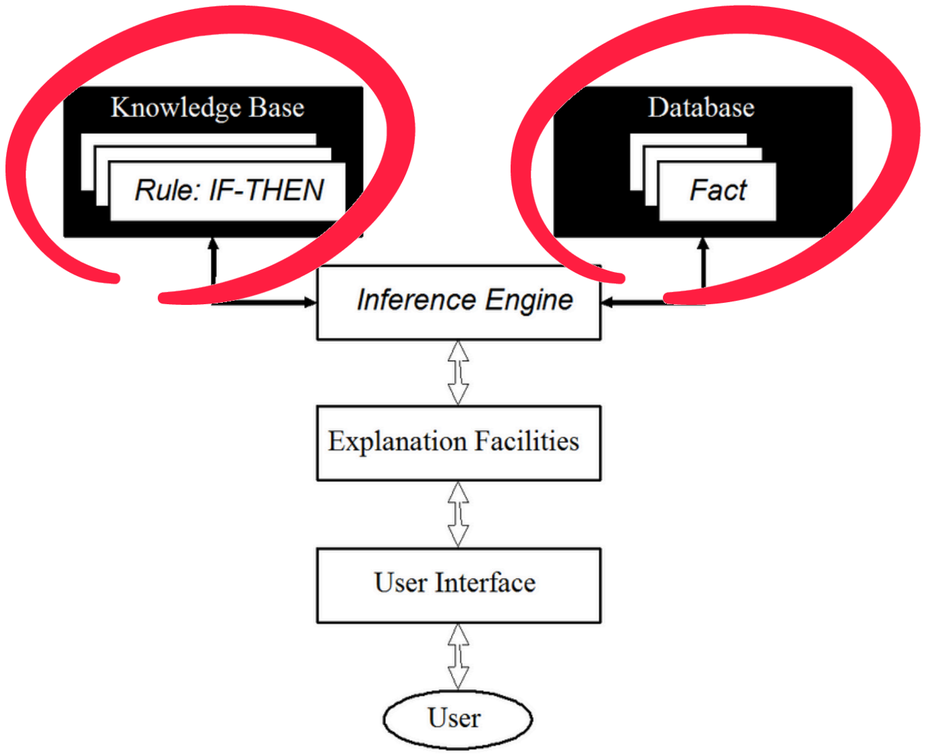
- *Rule 2*

  IF the patient's temperature is >100,
  THEN the patient has a fever.

- *Rule 3*

  IF the patient has been sick over a month
  AND the patient has a fever,
  THEN suspected a bacterial infection.

Observed facts:
1. Temperature of 102
2. Been sick for 2 months
3. Has a sore throat



EXAMPLE

FORWARD-CHAINING

- The system takes each rule in turn and checks to see it its premises are listed in the working memory.
- When the system finds matches for all the premises, it places the rule's conclusion in the working memory.
- Thus, the system gains new problem information that it uses for further reasoning.
- When no rule emerges that can be fired, the system stops.

Observed facts:

1. Temperature of 102
2. Been sick for 2 months
3. Has a sore throat

Rules

❑ *Rule 1*

IF the patient has a sore throat
AND suspected a bacterial infection,
THEN the patient has a strep throat.

❑ *Rule 2*

IF the patient's temperature is >100,
THEN the patient has a fever.

❑ *Rule 3*

IF the patient has been sick over a month
AND the patient has a fever,
THEN suspected a bacterial infection.

EXAMPLE

FORWARD-CHAINING | FCFS RULE

Cycle 1

- Rule 1, Premise 1
  - Sore throat? True
- Rule 1, Premise 2
  - Bacterial infection? Unknown
- Rule 2, Premise 1
  - Temperature >100? True
  - Fire Rule 2: Patient has fever

Rules

❑ *Rule 1*
IF the patient has a sore throat
AND suspected a bacterial infection,
THEN the patient has a strep throat.

❑ *Rule 2*
IF the patient's temperature is >100,
THEN the patient has a fever.

❑ *Rule 3*
IF the patient has been sick over a month
AND the patient has a fever,
THEN suspected a bacterial infection.

Observed facts:
✓ Temperature of 102
✓ Been sick for 2 months
✓ Has a sore throat

New observed facts:
✓ Patient has fever

EXAMPLE

FORWARD-CHAINING

Cycle 2: Rule 1 unknown and Rule 2 fired

- Rule 1……
- Rule 3, Premise 1
  - Sick over a month? True
- Rule 3, Premise 2
  - Fever? True
  - Fire Rule 3: Bacterial infection

Rules

❑ *Rule 1*
IF the patient has a sore throat
AND suspected a bacterial infection,
THEN the patient has a strep throat.

❑ *Rule 2*
IF the patient's temperature is >100,
THEN the patient has a fever.

❑ *Rule 3*
IF the patient has been sick over a month
AND the patient has a fever,
THEN suspected a bacterial infection.

Observed facts:
✓ Temperature of 102
✓ Been sick for 2 months
✓ Has a sore throat

New observed facts:
✓ Patient has fever
✓ Bacterial infection

EXAMPLE

FORWARD-CHAINING

Cycle 3

- Rule 1, Premise 2
  - Bacterial infection? True
  - Fire Rule 1: Step throat
- STOP

Rules

❑ *Rule 1*
IF the patient has a sore throat
AND suspected a bacterial infection,
THEN the patient has a strep throat.

❑ *Rule 2*
IF the patient's temperature is >100,
THEN the patient has a fever.

❑ *Rule 3*
IF the patient has been sick over a month
AND the patient has a fever,
THEN suspected a bacterial infection.

Observed facts:
✓ Temperature of 102
✓ Been sick for 2 months
✓ Has a sore throat

New observed facts:
✓ The patient has a strep throat

Conclusion: Patient infected with strep throat since no more rule can be fired

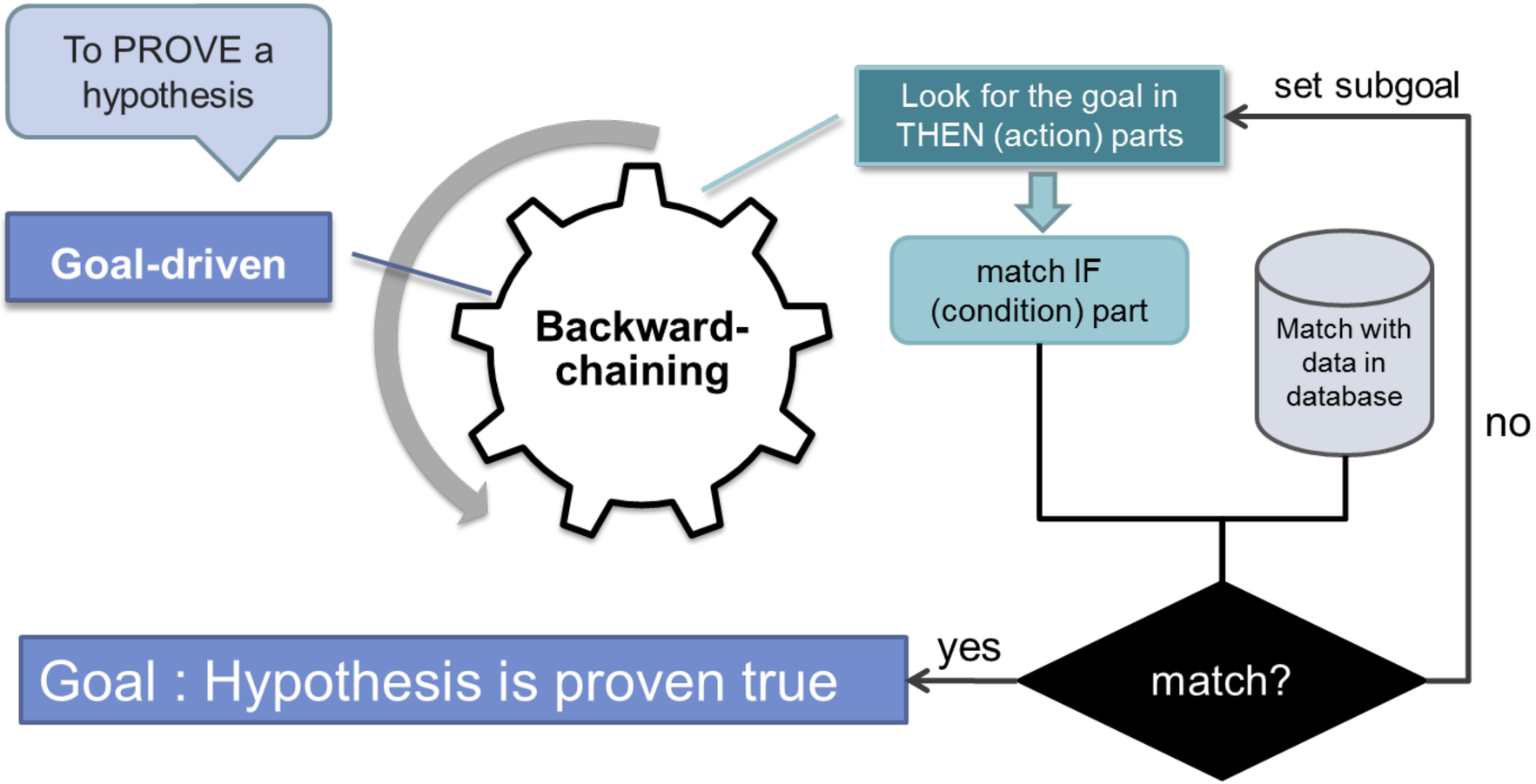
LIMITATIONS

FORWARD-CHAINING

1. We are required to gather all information which seems may be necessary
2. However, many rules executed may have nothing to do with the established goal.

INFERENCE MECHANISM

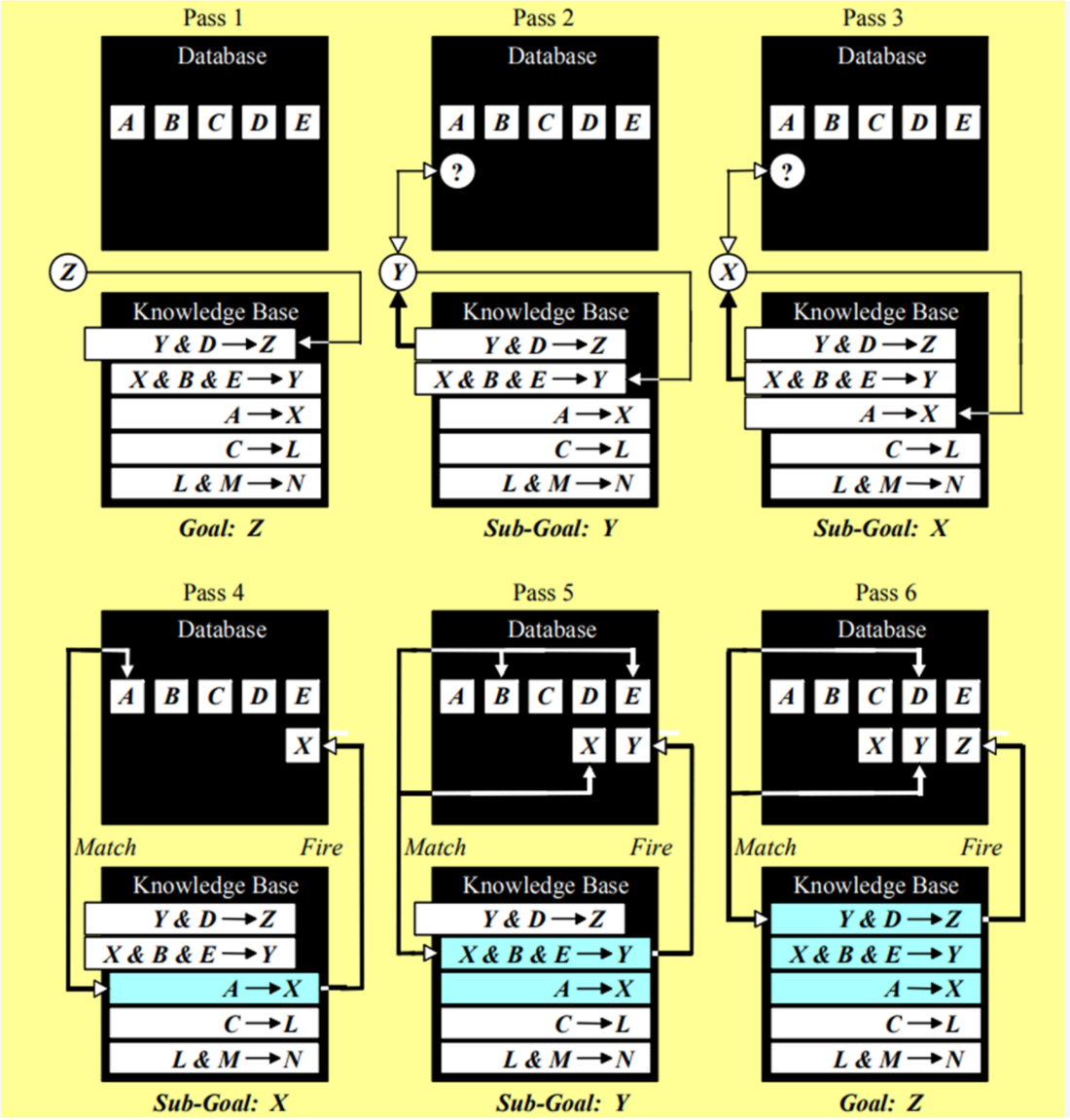
BACKWARD CHAINING



The match-fire cycle stops when no further rules can be fired to prove the (sub)goal(s).

INFERENCE MECHANISM

BACKWARD CHAINING



EXAMPLE

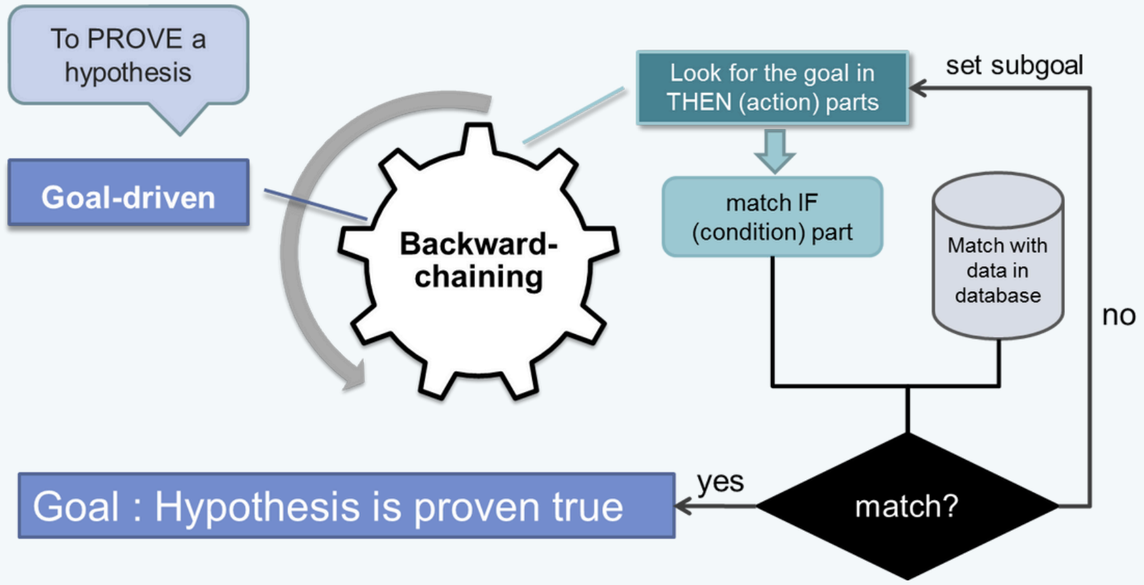
BACKWARD CHAINING

The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

2 facts about Fritz

1. Fritz croaks
2. Fritz eats flies



EXAMPLE

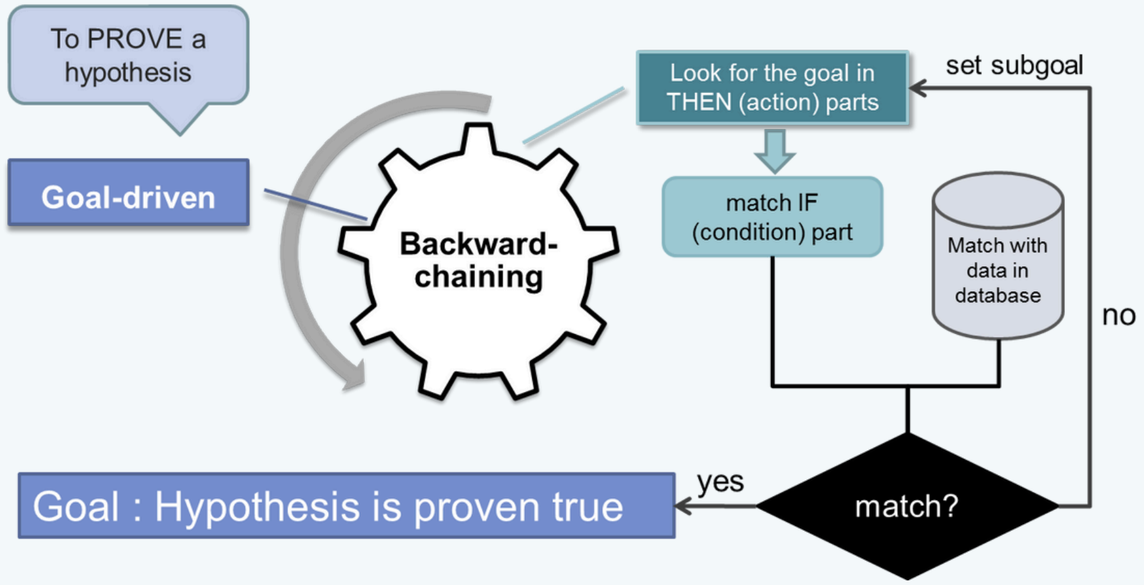
BACKWARD CHAINING

The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

2 facts about Fritz

1. Fritz croaks
2. Fritz eats flies



EXAMPLE

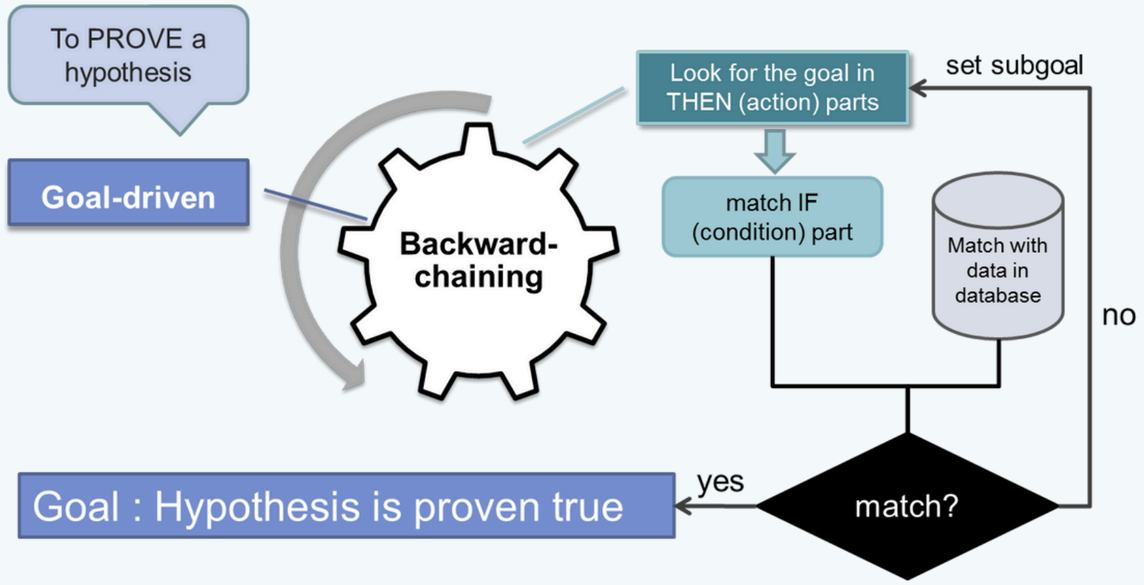
BACKWARD CHAINING

The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

2 facts about Fritz

1. Fritz croaks
2. Fritz eats flies



EXAMPLE

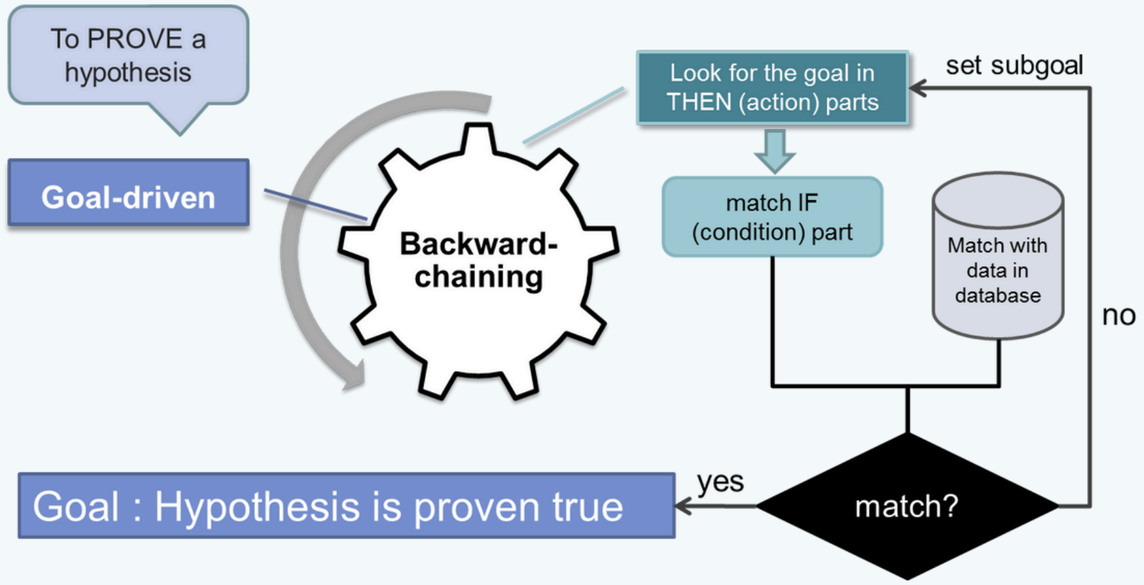
BACKWARD CHAINING

The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

2 facts about Fritz

1. Fritz croaks
2. Fritz eats flies



EXAMPLE

BACKWARD CHAINING

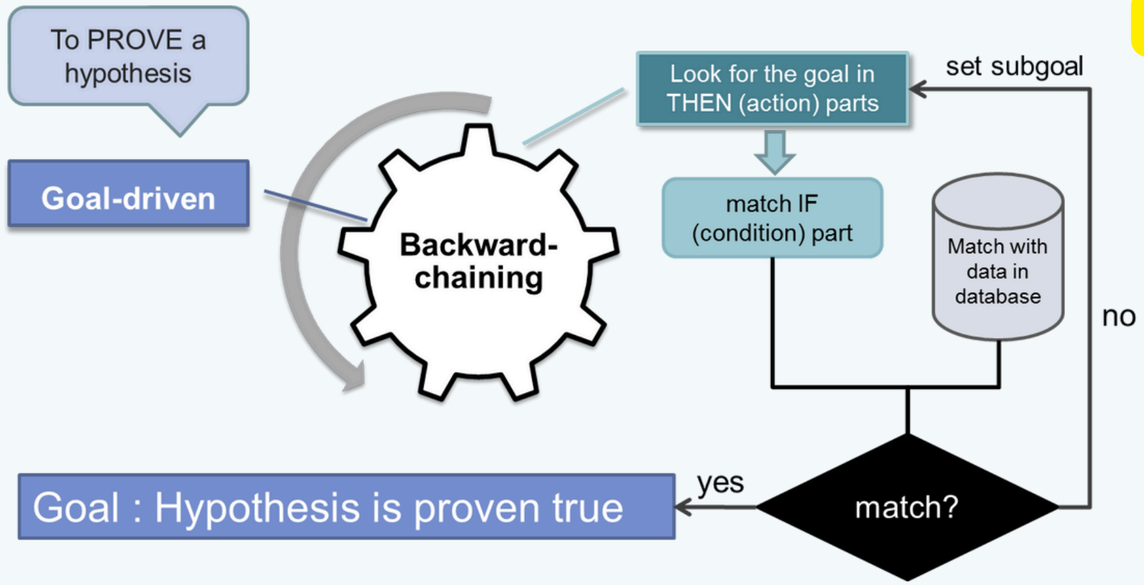
The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

2 facts about Fritz

1. Fritz croaks
2. Fritz eats flies

Therefore, Fritz is green is True



HOW DO WE CHOOSE?

Forward chaining

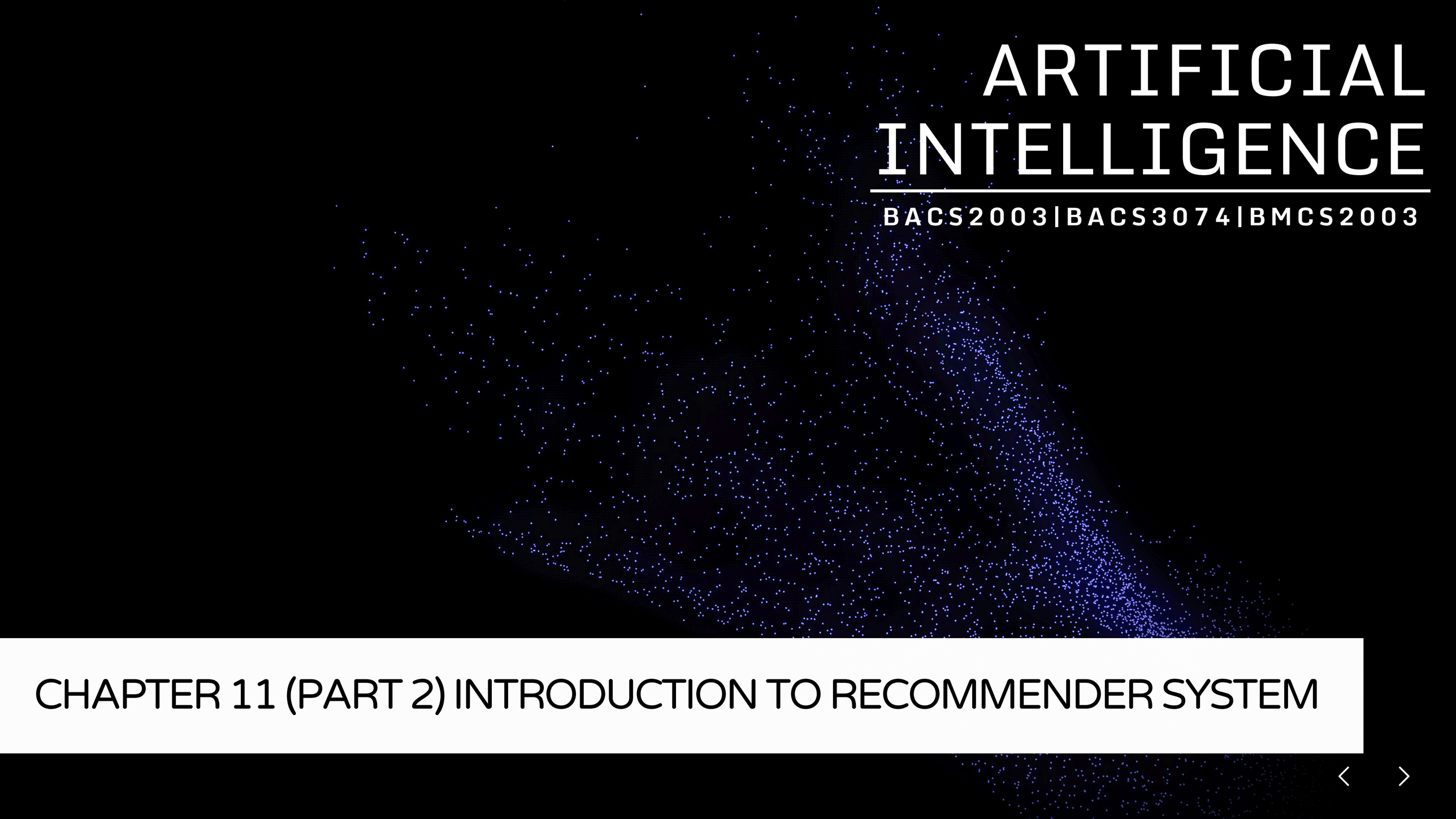
- needs to gather facts
- then only can try to infer from it
- e.g. identify a strange creature

Backward chaining

- begins with a hypothetical solution
- then attempts to find facts to prove it
- Faster search because goal-based
- e.g. Is it going to rain today?

ARTIFICIAL INTELLIGENCE

BACS2003|BACS3074|BMCS2003



CHAPTER 11 (PART 2) INTRODUCTION TO RECOMMENDER SYSTEM

OUTCOMES

1. **Introduction**
2. **Types of Recommender Systems – Collaborative Filtering and Content-based Filtering**
3. **Matrix Factorization**

INTRODUCTION

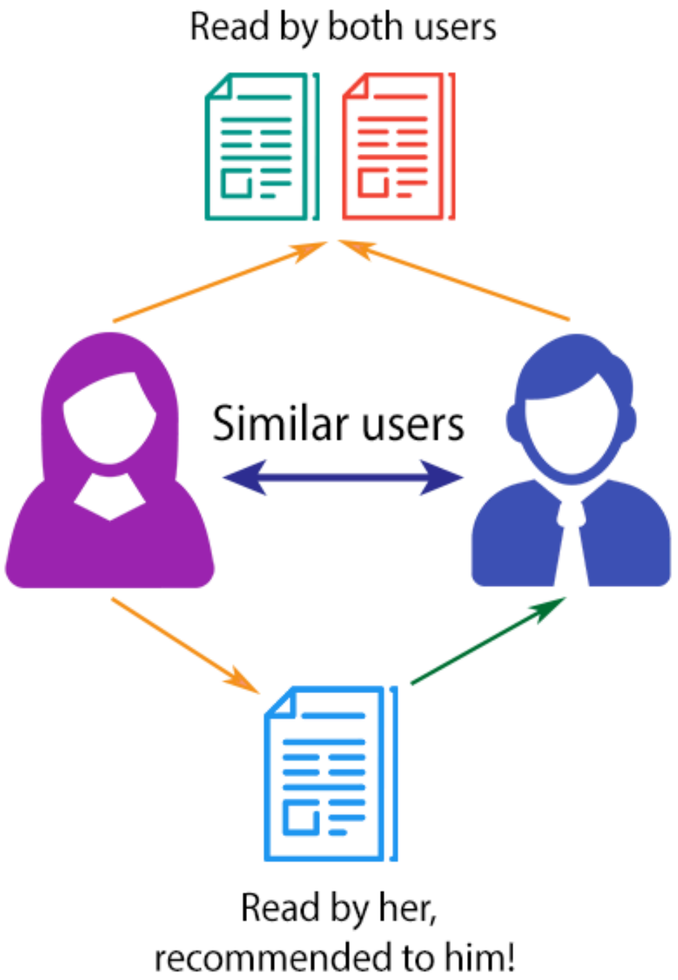
- Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user.
- Recommendations are usually personalized, different users or user groups receive diverse suggestions.

ADVANTAGES

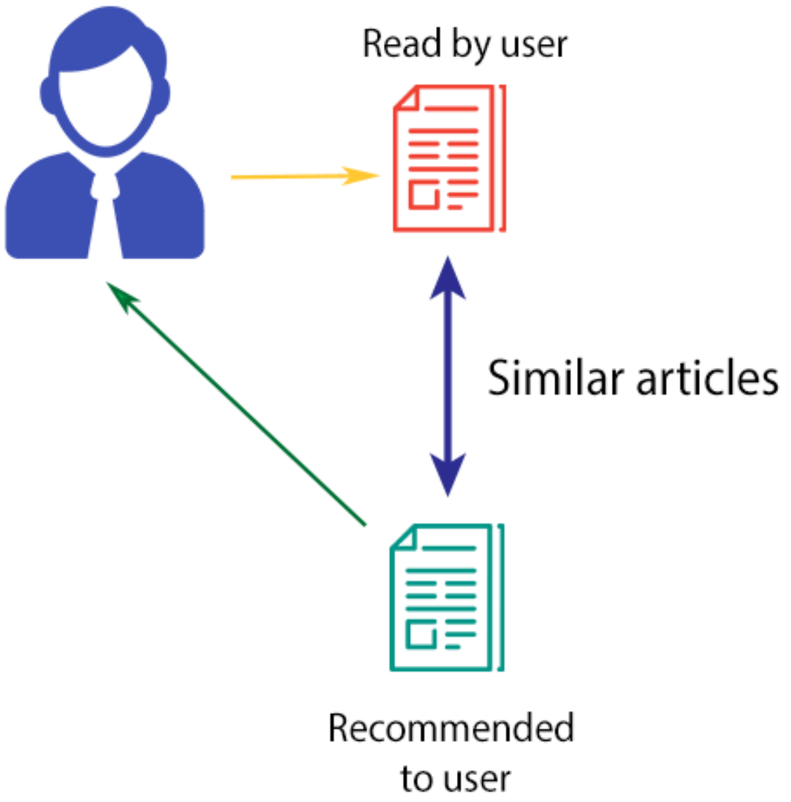
1. Increased Revenue and Sales
2. Personalization
3. Improved Customer Engagement
4. Increased Traffic
5. Customer Retention
6. Increased Average Order Value
7. Increased Number of Items per Order
8. Adaptive
9. Promotion of Less Popular Products

COMMON TYPES OF RS

COLLABORATIVE FILTERING



CONTENT-BASED FILTERING



CONTENT-BASED FILTERING

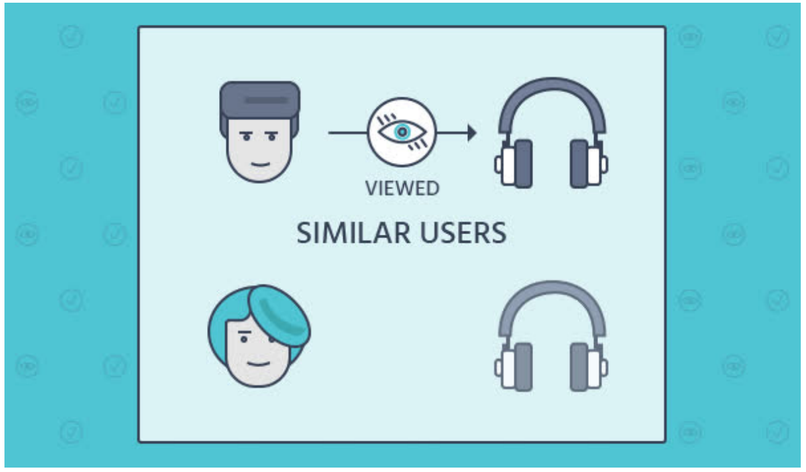
- Content-based filtering approaches utilize a series of discrete, pre-tagged characteristics of an item in order to recommend additional items with similar properties.
- Search- or content-based methods treat the recommendations problem as a search for related items.
- The algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects.
- If the user has few purchases or ratings, search based recommendation algorithms scale and perform well.

LIMITATION OF CONTENT-BASED FILTERING

- Limited in scope, for example, it can only make recommendations that are similar to the original seed.
- For users with thousands of purchases, however, it’s impractical to base a query on all the items.
- The algorithm must use a subset or summary of the data, reducing quality.
- In all cases, recommendation quality is relatively poor.
- Recommendations should help a customer find and discover new, relevant, and interesting items.

COLLABORATIVE FILTERING

- Collaborative filtering approaches build a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in.
- The system generates recommendations using only information about rating profiles for different users or items.



STRENGTH

- A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.

LIMITATIONS

- Requires a large amount of information about a user to make accurate recommendations (Cold start: For a new user or item, there isn't enough data make accurate recommendations)
- Scalability – big data (items and users amount)
- Sparsity – how many of you actually rated a product?

DATA COLLECTION

- Explicit Data Collection
- Implicit Data Collection

EXPLICIT DATA COLLECTION

Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale.
- Asking a user to search.
- Asking a user to rank a collection of items from favorite to least favorite.
- Presenting two items to a user and asking him/her to choose the better one of them.
- Asking a user to create a list of items that he/she likes.

IMPLICIT DATA COLLECTION

Examples of explicit data collection include the following:

- Observing the items that a user views in an online store.
- Analyzing item/user viewing times.
- Keeping a record of the items that a user purchases online.
- Obtaining a list of items that a user has listened to or watched on his/her computer.
- Analyzing the user's social network and discovering similar likes and dislikes.

MODELING APPROACH

COLLABORATIVE FILTERING

USER-BASED

Collaborative filtering and cluster models focus on finding similar set of customers whose purchased and rated items overlap with the user's purchased and rated items.

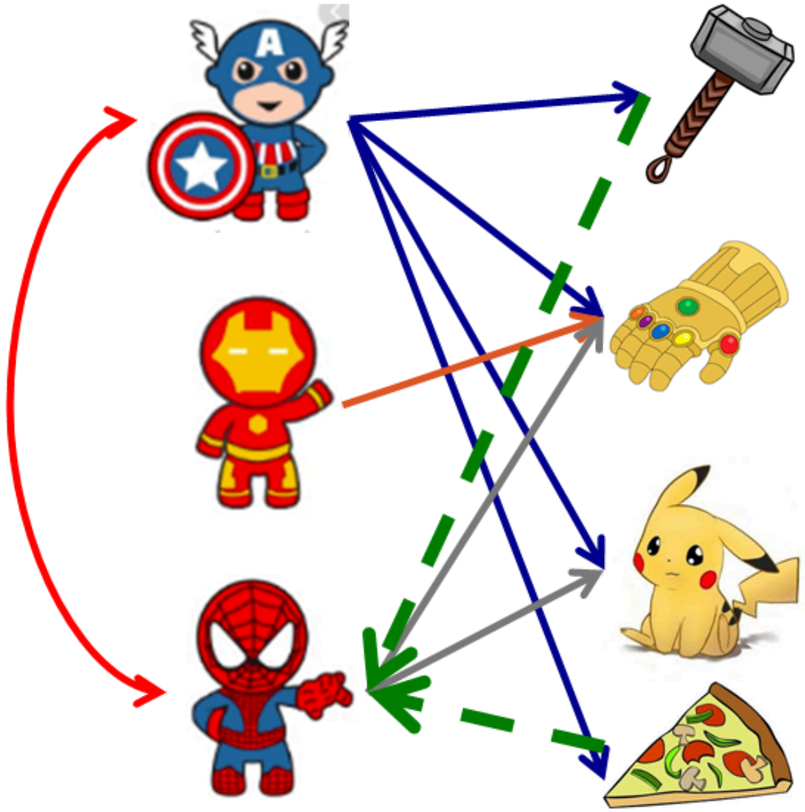
ITEM-BASED

Search-based methods and item-to-item collaborative filtering focus on finding similar items, not similar customers.

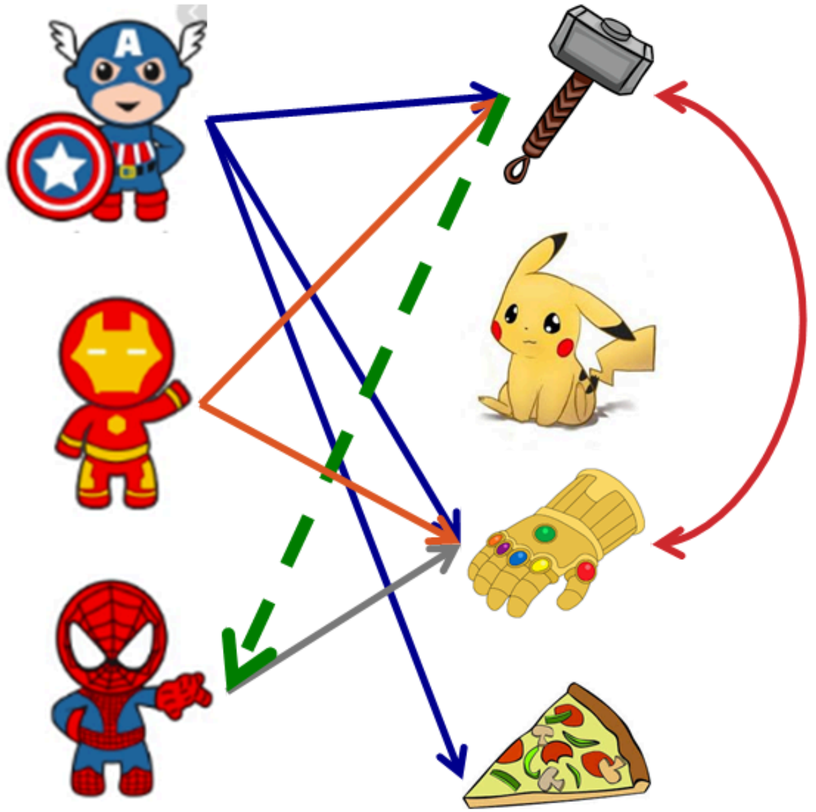
MODELING APPROACH

COLLABORATIVE FILTERING

USER-BASED



ITEM-BASED



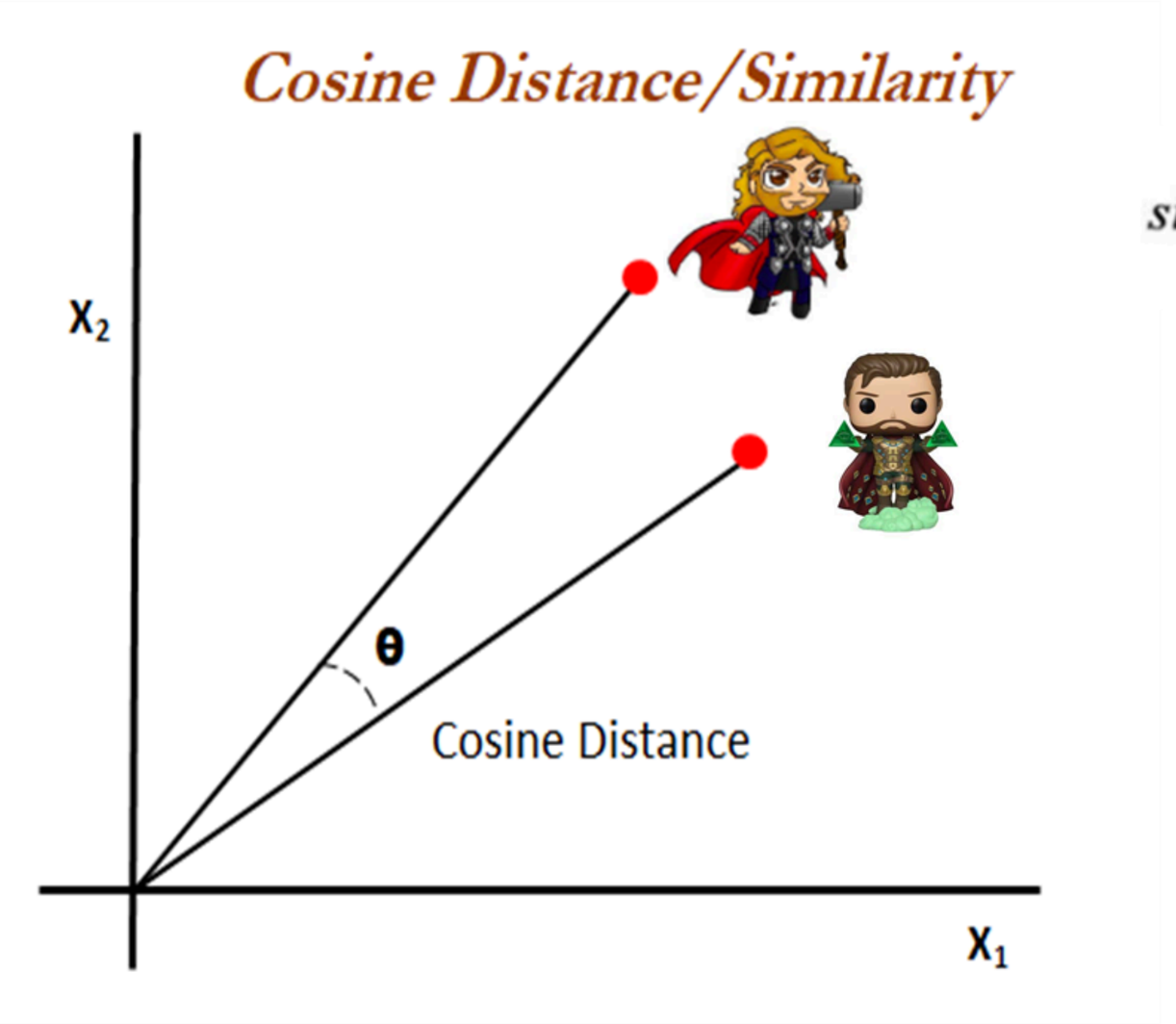
MODELING APPROACH

COLLABORATIVE FILTERING

USER-BASED	ITEM-BASED
items usually don't change much, and item based approach often can be computed offline and served without constantly re-training	**User-based approach is often harder to scale because of the dynamic nature of users (Geo-dependent interests)**

Item based approach is usually preferred over user-based approach.

USER-BASED COLLABORATIVE FILTERING



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

LIMITATION

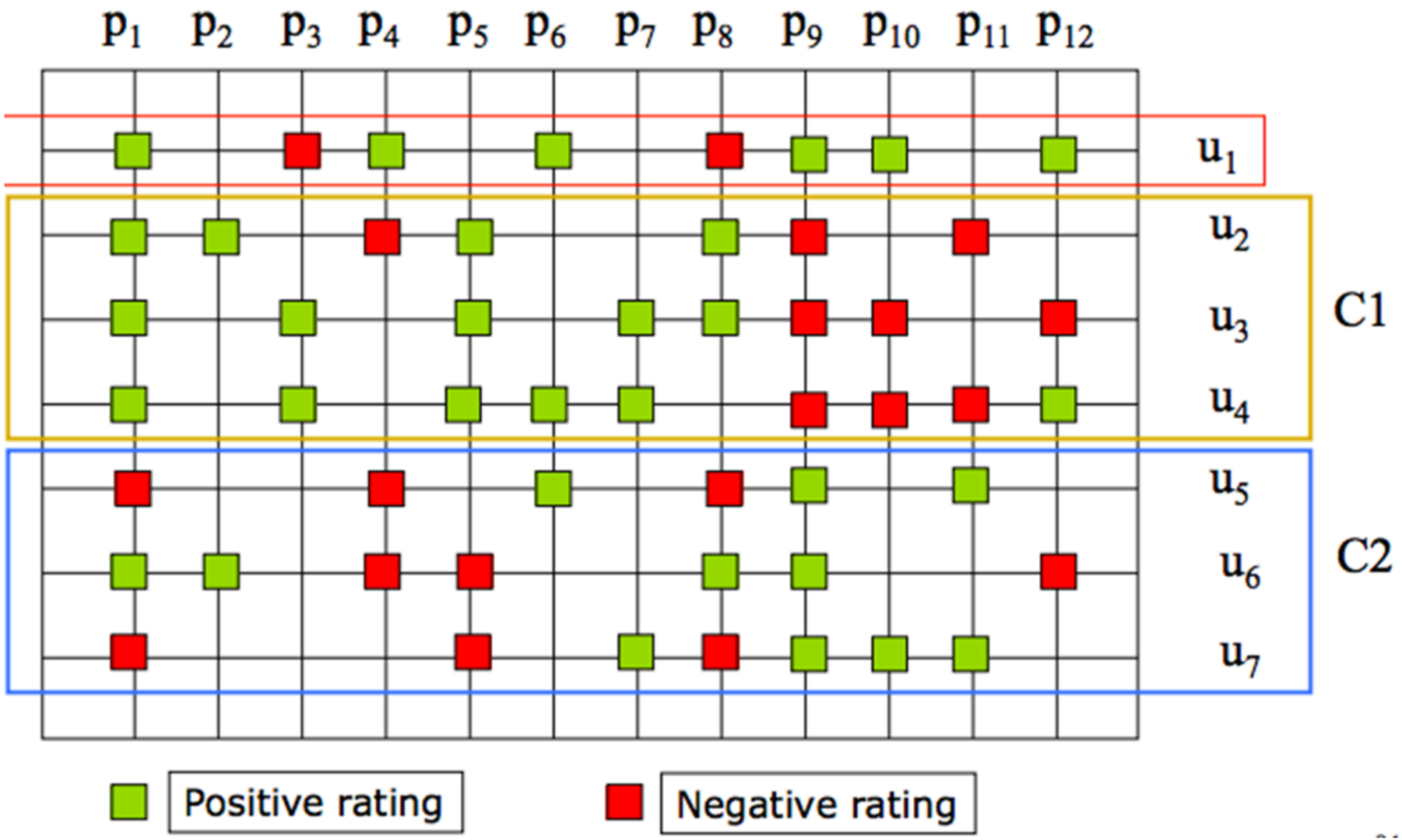
- Severe performance and scaling issues.
- Worst case complexity is O(MN)[M- Number of customers, N - Number of product catalog].

USER-BASED CLUSTERING

- Divide the customer base into many segments and assign the user to the segment containing the most similar customers.
- The algorithm’s goal is to assign the user to the segment containing the most similar customers. It then uses the purchases and ratings of the customers in the segment to generate recommendations.

EXAMPLE

USER-BASED CLUSTERING



PROS & CONS

- Cluster models have better online scalability and performance than collaborative filtering because they compare the user to a controlled number of segments rather than the entire customer base.
- Recommendations produced could be less relevant.
- Online user–segment classification becomes almost as expensive as finding similar customers using collaborative filtering.

ITEM-TO-ITEM COLLABORATIVE FILTERING

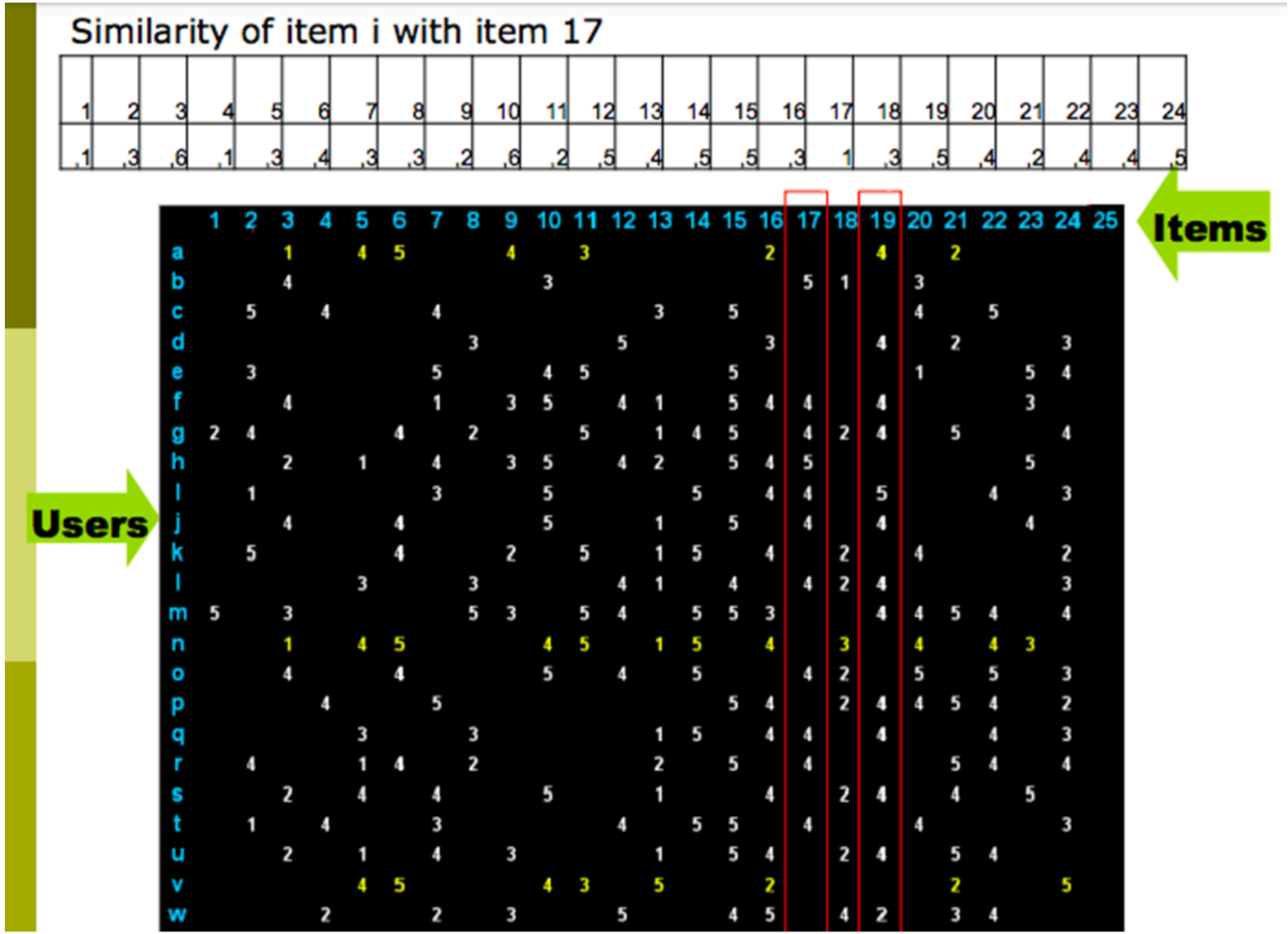
- Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list.
- To determine the most-similar match for a given item, the algorithm builds a similar-items table by finding items that customers tend to purchase together.
- Given a similar-items table, the algorithm finds items similar to each of the user's purchases and ratings, aggregates those items, and then recommends the most popular or correlated items. This computation is very quick, depending only on the number of items the user purchased or rated.

SIMILAR ITEMS

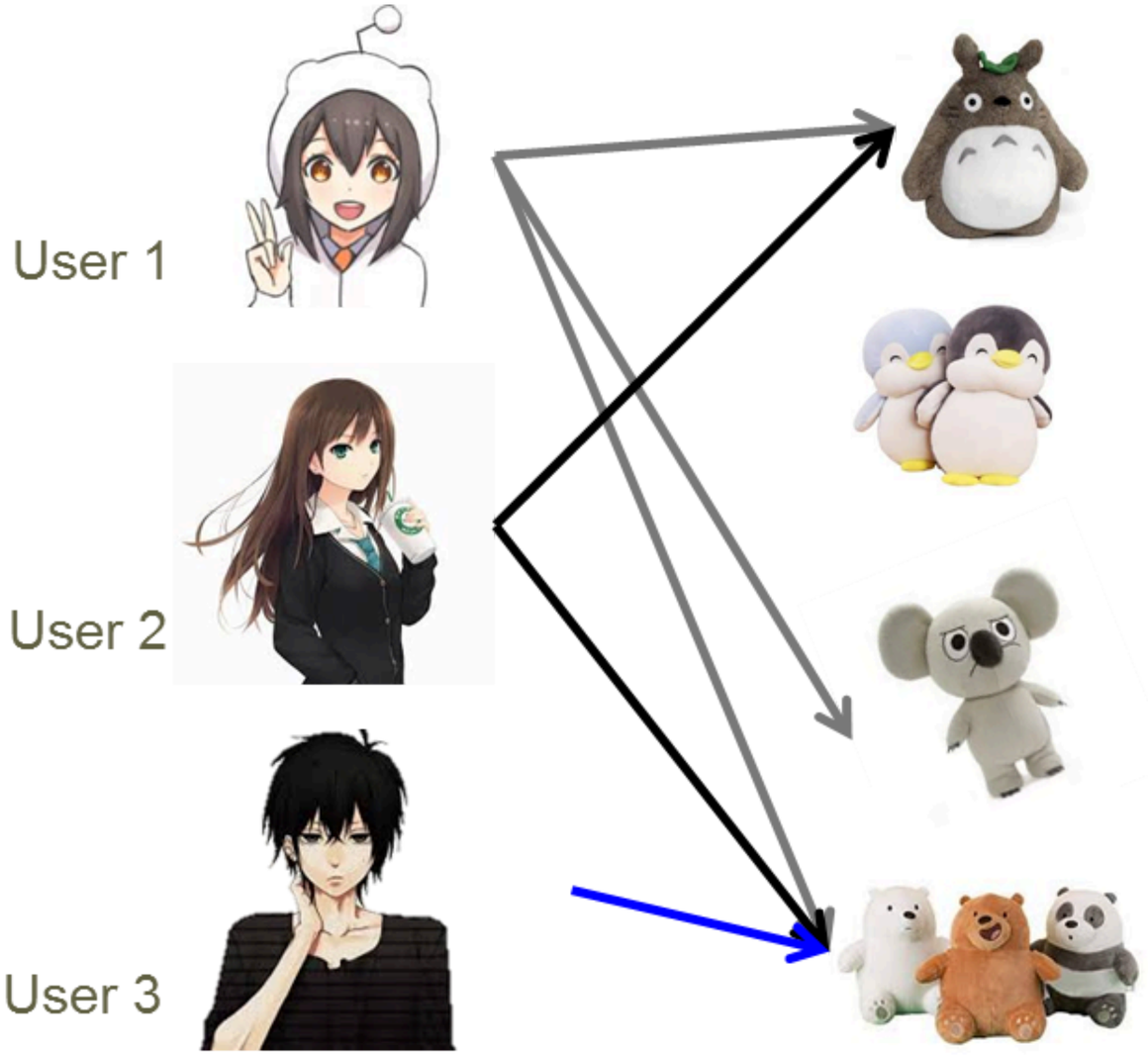
Similarity can be computed in a number of ways

- Using the user ratings
- Using some product description
- Using co-occurrence of items in a bag or in the set of a user past purchased products

EXAMPLE


Similarity of item i with item 17
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
,1 ,3 ,6 ,1 ,3 ,4 ,3 ,3 ,2 ,6 ,2 ,5 ,4 ,5 ,5 ,3 1 ,3 ,5 ,4 ,2 ,4 ,4 ,5
Items
Users

EXAMPLE



What item would the RS recommends to User 3?

PROS

For item-to-item it is dependent only on how many items the user has purchased or rated. Thus, the algorithm is fast even for extremely large data sets. Because the algorithm recommends highly correlated similar items, recommendation quality is excellent.

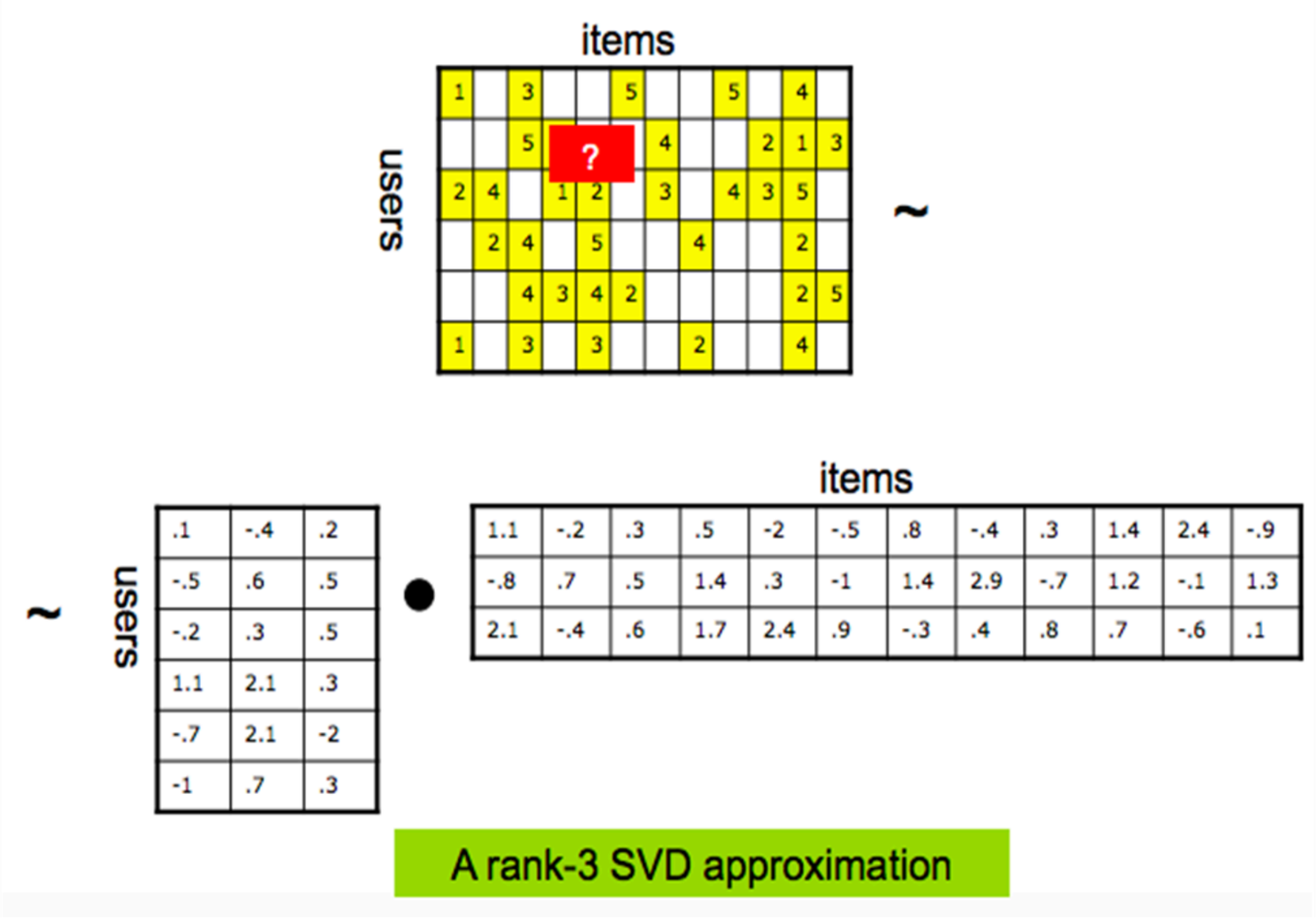
MATRIX FACTORIZATION

	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

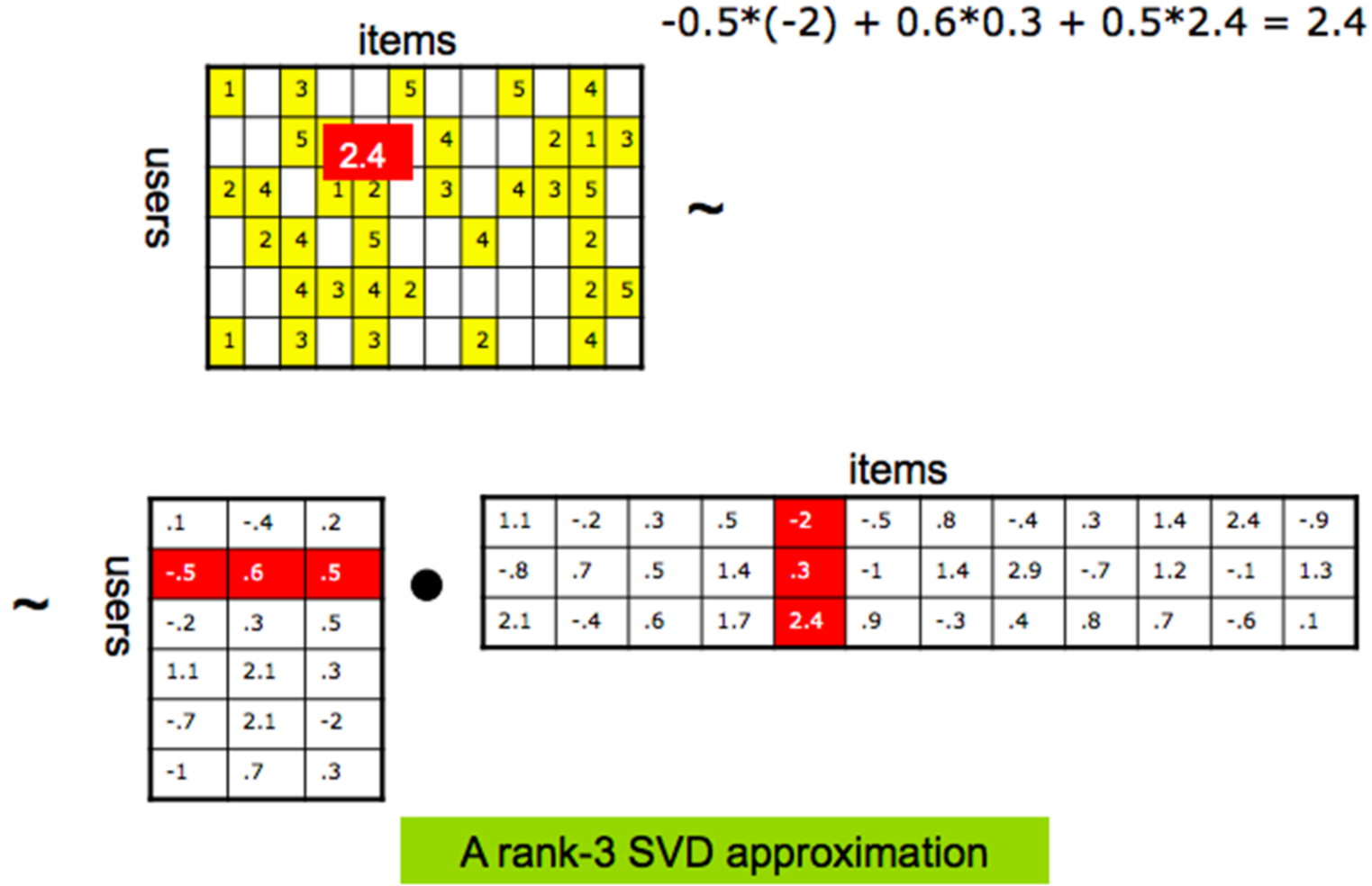
MATRIX FACTORIZATION

Estimate Unknown Ratings



MATRIX FACTORIZATION

Estimate Unknown Ratings



THE END

NEXT LECTURE

Probability and Certainty Factor