ADVANCED KNOWLEDGE ENGINEERING

Instructor:

KHUAT Thanh Tung

University of Technology Sydney

Subject's Outline

- **Topic 1:** An Overview of Knowledge Engineering
- **Topic 2:** An Overview of Knowledge-based Systems
- **Topic 3:** Knowledge Acquisition
- **Topic 4:** Knowledge Representation and Reasoning

Mid-term assessment

- **Topic 5:** Ontology
- **Topic 6:** Knowledge Graphs
- **Topic 7:** Expert Systems
- **Topic 8:** Uncertain Reasoning
- **Topic 9:** Hybrid Knowledge-based Systems
- **Topic 10:** Automated AI Planning

Group projects for the advanced topics

Expert Systems and Related Issues

TOPIC 7

Objectives of this topic

By the end of this topic, you will be able to:

- ✓ know the definition of expert systems
- ✓ learn how to build expert systems using Shells and PROLOG.
- ✓ know basic knowledge of PROLOG programming language

Expert Systems

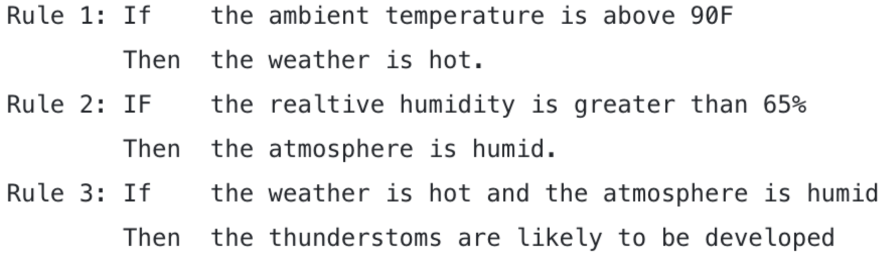
- ❑ What is an expert system?
  - ❑ An intelligent program that can mimic the problem-solving behavior of a human expert
  - ❑ a computer program that symbolizes the knowledge of an expert in a certain domain
    - ❑ Humans knowledge consists of subject-specific knowledge/ domain knowledge and problem-solving knowledge

Expert Systems

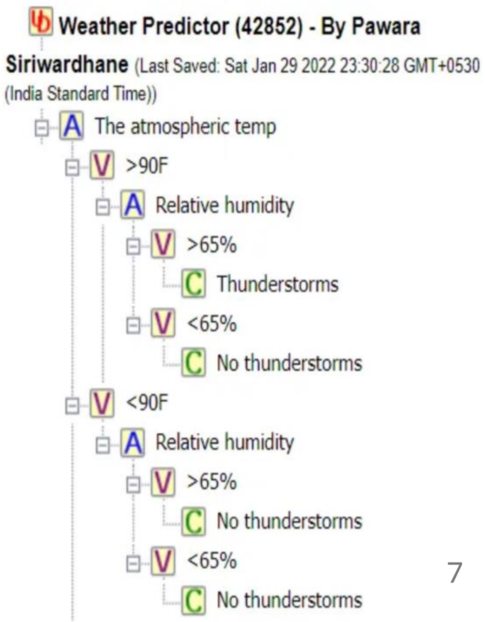
- ❑ Key features of an expert system
  - ❑ Operates on a specific domain
  - ❑ Dominates the question asking process
  - ❑ Process incomplete information
  - ❑ Provide certainty of an answer given by the expert system
  - ❑ Process alternative solutions
  - ❑ Provide reasons for answers
- ❑ ES can be developed from scratch (using Logic Programming language like **ProLog**) or can easily be implemented by using an expert system shell

Expert System shells

- ❑ Toolkits can be used to develop expert systems
  - ❑ some built expert system components with an empty knowledge base
  - ❑ a software package facilitates the building of knowledge-based expert systems by providing a knowledge representation scheme and an inference engine
- ❑ Difference ES shells offer various ways to model the knowledge into the knowledge base
  - ❑ As rules
  - ❑ in the form of a decision tree
  - ❑ as objects (frames) - A data structure with typical knowledge about a particular object or concept



```
Rule 1: If    the ambient temperature is above 90F
        Then  the weather is hot.
Rule 2: IF    the realtive humidity is greater than 65%
        Then  the atmosphere is humid.
Rule 3: If    the weather is hot and the atmosphere is humid
        Then  the thunderstoms are likely to be developed
```



Weather Predictor (42852) - By Pawara

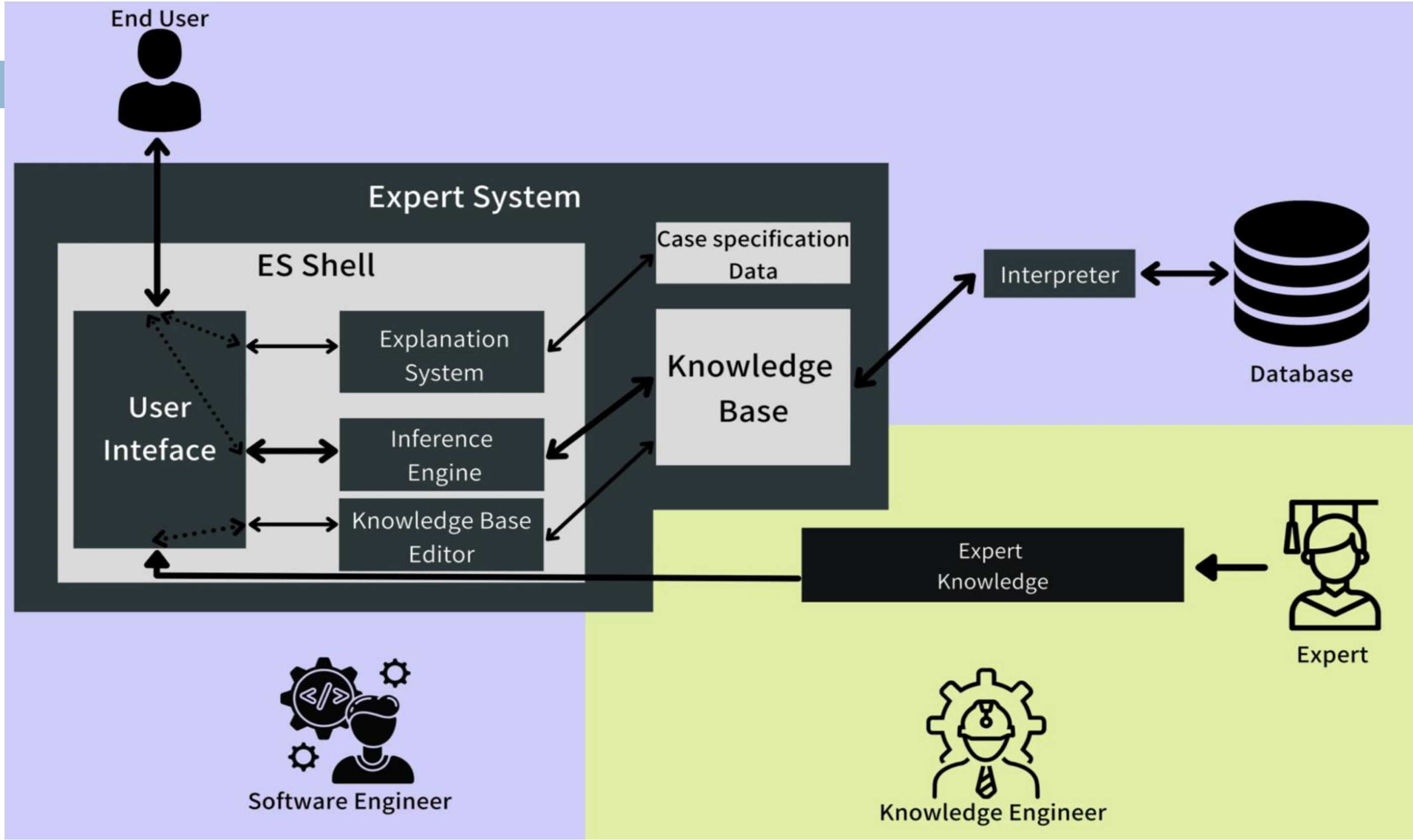
Siriwardhane (Last Saved: Sat Jan 29 2022 23:30:28 GMT+0530 (India Standard Time))

- A The atmospheric temp
  - V >90F
    - A Relative humidity
      - V >65%
        - C Thunderstorms
      - V <65%
        - C No thunderstorms
  - V <90F
    - A Relative humidity
      - V >65%
        - C No thunderstorms
      - V <65%
        - C No thunderstorms

Expert System Shell Structure

- The Expert System Shell refers to a software module containing an:
  - User interface (built-in)
  - Inference engine (built-in)
  - A structured skeleton of a knowledge base (in its empty state) with the suitable knowledge representation facilities
  - some ES Shells: provide facilities for database connectivity through interpreter, web integration, and natural language processing features
- The user interface
  - the portal available for both
    - end-users (use the expert system to get solutions)
    - knowledge engineers (perform the knowledge engineering and modelling)

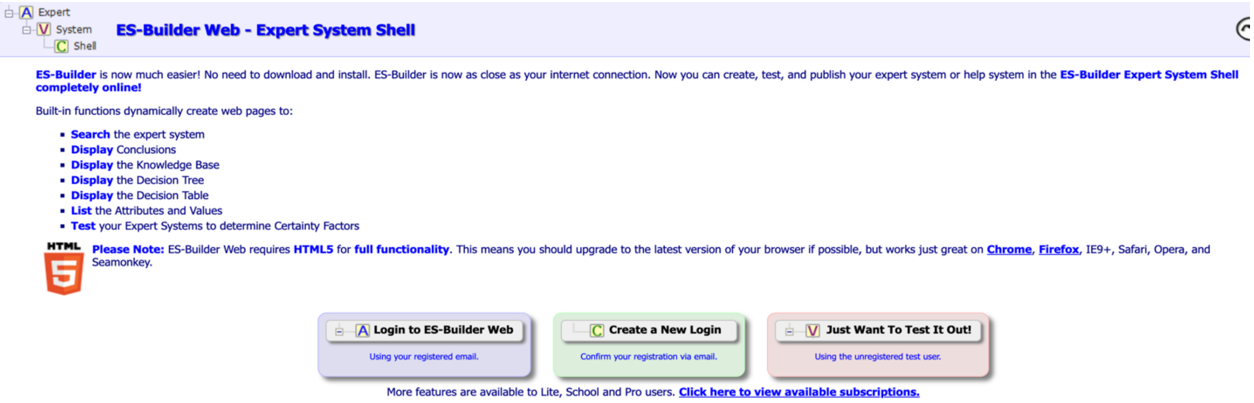
Expert System Shell Structure



- ❑Inference engine: most important part of an ES
  - ❑access the knowledge base and solves the problem
    - ❑backward chaining or forward chaining of facts and rules in the knowledge base
  - ❑a built-in component that is usually programmed in ProLog

Popular Expert System Shells

- ❑ ES-Builder
  - ❑ especially for students and researchers to develop expert system shells
  - ❑ an improved web interface built using the AJAX framework
  - ❑ stores the facts and rules of the knowledge base in an online MySQL database
  - ❑ a built-in inference engine (written in Prolog)and user interfaces are developed using simple HTML and CSS
  - ❑ The rule base knowledge base can also be developed using a decision tree



ES-Builder Web - Expert System Shell

ES-Builder is now much easier! No need to download and install. ES-Builder is now as close as your internet connection. Now you can create, test, and publish your expert system or help system in the **ES-Builder Expert System Shell completely online!**

Built-in functions dynamically create web pages to:

- **Search** the expert system
- **Display** Conclusions
- **Display** the Knowledge Base
- **Display** the Decision Tree
- **Display** the Decision Table
- **List** the Attributes and Values
- **Test** your Expert Systems to determine Certainty Factors

Please Note: ES-Builder Web requires **HTML5** for **full functionality**. This means you should upgrade to the latest version of your browser if possible, but works just great on **Chrome**, **Firefox**, IE9+, Safari, Opera, and Seamonkey.

HTML 5
Login to ES-Builder Web
Using your registered email.
Create a New Login
Confirm your registration via email.
Just Want To Test It Out!
Using the unregistered test user.

More features are available to Lite, School and Pro users. **Click here to view available subscriptions.**

Popular Expert System Shells

- ❑ CLIPS
  - ❑ C-Language Integrated Production System
  - ❑ written in the procedural langue C
  - ❑ developed in 1985 at NASA's Johnson Space Center
  - ❑ a rule-based programming language
  - ❑ used in systems where the heuristic solution is easier to implement and maintain than a traditional algorithmic approach
  - ❑ provides 3 different tools for knowledge representation in the form of programming methodologies
    - ❑ Procedural
    - ❑ Object-oriented
    - ❑ Rule-based programming
  - ❑ The expert system developed by CLIPS requires ANSI compiler

Popular Expert System Shells

- PyKE
  - Python Knowledge Engine
  - uses logic programming that is inspired by Prolog, but PyKE is entirely written in the programming language Python
  - major features of PyKE knowledge base
    - python functions
    - PyKE rules
    - PyKE pattern variables
    - graph plans
  - an inference engine uses rules and facts to create additional facts using forward chaining of rules to prove goals
  - through backward chaining, it assembles Python functions into customizable call graphs which are also known as Plans

Expert System Programming Languages

- ❑ Expert System Programming Languages
  - ❑ Procedural languages
    - ❑ C++, Java,…
    - ❑ general-purpose languages
    - ❑ be organised as a set of procedures very similar to the way that a chapter in a book is divided into a series of paragraphs
    - ❑ offer NO specific support for the development of ESs
  - ❑ Declarative programming language PROLOG
    - ❑ specifically designed for programming AI systems from scratch
    - ❑ allow more flexibility for developing the ES component
    - ❑ it has limited inbuilt facilities, i.e., less specific support for the development of ESs
    - ❑ developing an ES using PROLOG will take considerably longer than if ES shells were used.

An Introduction to PROLOG

- Programs written in declarative languages include a set of declarations about a specific field of knowledge
  - Using this declaration, the ES can determine the truth of a statement as well as work out solutions to problems
- Give knowledge to an ES in the form of facts
- PROLOG programs are made from **terms**
  - A constant
    - a single entity (like cat, 'Bob') or a non-negative integer
    - constants **cannot** begin with a **capital** letter unless they are enclosed in quotes
  - A variable
    - a series of letters that begins with a capital letter (like Brian)
  - A structure
    - a predicate with zero or more arguments, written in functional notation

An Introduction to PROLOG

- An example for a PROLOG program

```
animal(zebra) .
speaks(boris, english) .
```

  - A fact is a term followed by a period (.)
  - A rule is a term followed by :- and a series of terms (term1, term2, . . . , termN) separated by commas (,) and ended by a period (.)
  - term :- term1, term2, ..., termN .

An Introduction to PROLOG

❑A PROLOG program is **a series of facts and rules**

❑In all situations the activity is placed before the brackets, which contain the objects/arguments affected by the activity/predicate.

```
speaks(boris, russian) .
speaks(john, english) .
speaks(mary, russian) .
speaks(mary, english) .
understands(Person1, Person2) :- speaks(Person1, L), speaks(Person2, L) .
```

This program can be translated into the following English facts

❑Boris speaks Russian.

❑ John speaks English.

❑ Mary speaks Russian.

❑ Mary speaks English.

and the following rule:

❑ Two people can understand each other if they both speak the same language.

An Introduction to PROLOG

- A PROLOG program is **a series of facts and rules**
  - the order of arguments within the brackets have no significance within PROLOG
    - But the order must be used consistently
  - For example, writing ‘the driver drives the car’ should always be written as:

```
drives(driver, car) .
```

Rather than sometimes as:

```
drives(driver, car) .
```

and at other times as:

```
drives(car, driver) .
```

An Introduction to PROLOG

- Expressing a fact in PROLOG
  - **Example:** A fact in English may be written as:

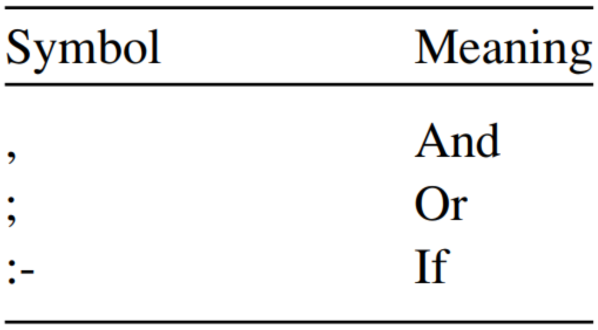
    The expert system monitors the ventilator.
  - This fact contains two important components:
    - a relationship or *predicate* in the PROLOG language. In this example, the predicate is *monitor.*
    - objects or *arguments* in PROLOG (objects are normally people, things or other items being acted on by the predicates). In this example, the objects are *expert system* and *ventilator*.
  - In PROLOG, the fact would be expressed (all in lower case) as:

    monitors(expert_system, ventilator) .

An Introduction to PROLOG

- Expressing a fact in PROLOG
  - The activity is placed at the beginning of the fact.
  - The people or objects affected by the activity appear inside the brackets.
    - normally with the person first, followed by any collective noun (e.g. class of pupils) or names of objects.
  - Note also that the syntax demands **full stops** at the **end**.

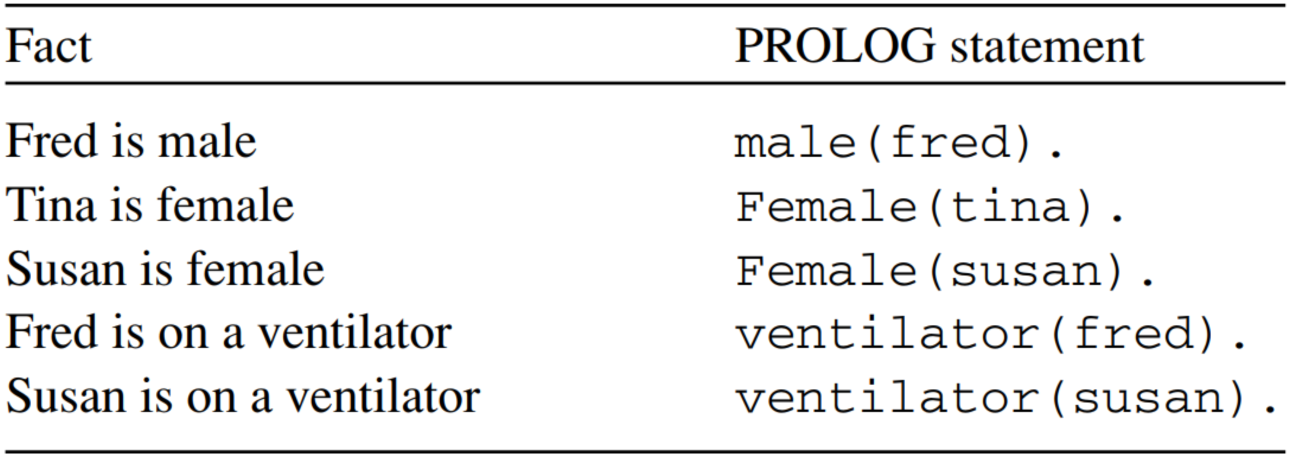
- PROLOG uses various symbols.



Symbol	Meaning
,	And
;	Or
:-	If

Using Facts in Expert Systems

- Given a set of facts, an ES can review those facts to determine if any apply in a given situation.
  - **Example:** a system can be provided with the following set of facts



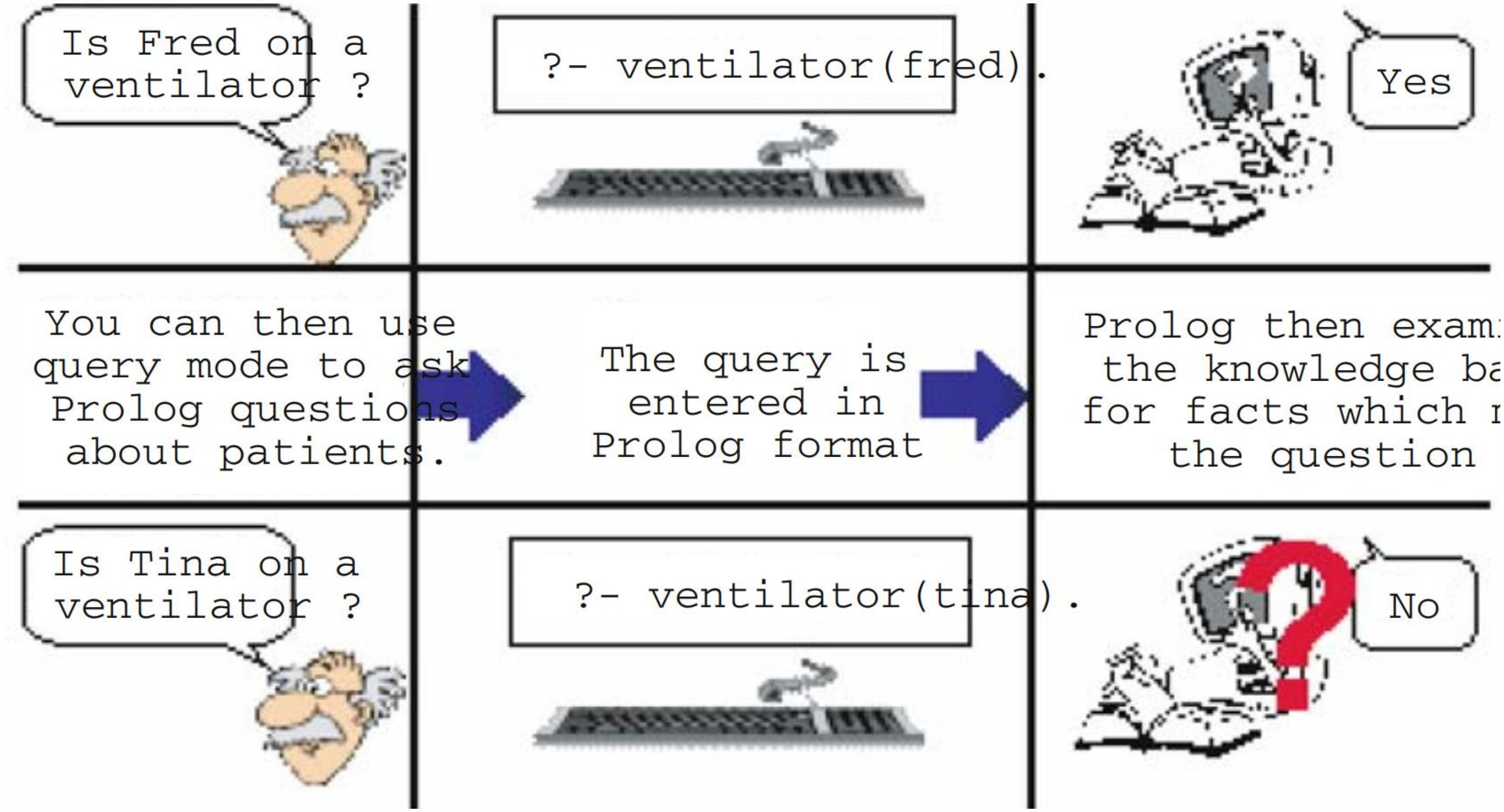
Fact	PROLOG statement
Fred is male	`male(fred).`
Tina is female	`Female(tina).`
Susan is female	`Female(susan).`
Fred is on a ventilator	`ventilator(fred).`
Susan is on a ventilator	`ventilator(susan).`

  - Queries can be given to PROLOG in the format:

    ? - ventilator(fred) .

    - ***Meaning***:please find out if Fred is on a ventilator
    - The ES then searches the knowledge base to see if this fact is known and a suitable response is provided.

Querying an Expert System



```
?- ventilator(fred).
```

You can then use query mode to ask Prolog questions about patients.

The query is entered in Prolog format

Prolog then exam the knowledge ba for facts which the question

```
?- ventilator(tina).
```

Querying an Expert System

❑ **Exercise**: Produce PROLOG statements for the facts listed below:

- Thomas is male
- Sally is female
- Bob is male
- Thomas has a single ticket
- Sally has a return ticket

What PROLOG query would you use to determine whether Thomas has a return ticket?

Extracting a Set of Records from an Expert System

- ❑The question structure within PROLOG
  - ❑can be used to identify and extract a set of related facts from the total of all facts given to PROLOG.
  - ❑If a variable is placed where a query is to be made about the facts, PROLOG then searches through the facts and returns any matches.
- ❑**Example:** which patients are female from the set of facts concerning patients used in previous example?
  - ❑The PROLOG statement will be written as

    ```
    ? - female(Patient) .
    ```

  - ❑ The **initial capital letter** in 'Patient' indicates that it is a **variable.**
  - ❑ All **variables** must *begin* with an **uppercase letter**.

Combining Queries

- ❑ In some situations, it will be necessary to extract records from two sets of different facts using PROLOG.
  - ❑ **Example:** names of some males and respiratory conditions are stored in the following facts

```
male(tim) .
male(marc) .
male(simon) .
resp(tim, acute) .
resp(marc, medium) .
resp(simon, acute) .
```

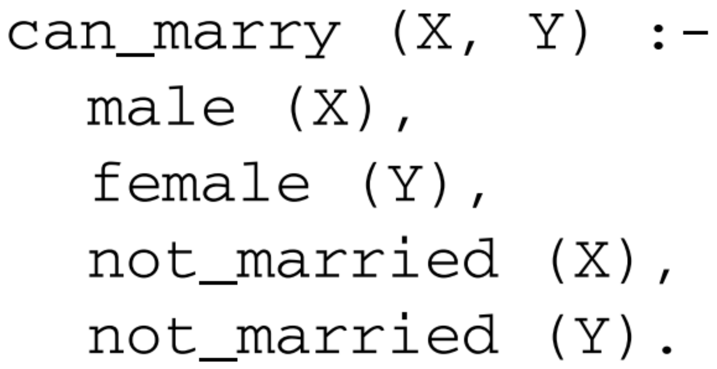
  - ❑ a reasonable question to ask the ES is:

    ‘Do any male patients have an acute respiratory condition?’
  - ❑ in PROLOG format:

```
? - male(X), resp(X, acute)
```

Inferences

- ❑PROLOG can perform backward-chaining inference from facts provided to it.
  - ❑**Example:** the following clause can be used to determine whether or not two people can marry:



```
can_marry (X, Y) :-
   male (X),
   female (Y),
   not_married (X),
   not_married (Y).
```

  - ❑From this information PROLOG can determine that two people can marry if X is male and Y is female, and if neither person is already married.

Working with Lists in PROLOG

- PROLOG provides a mechanism for working with lists.
- A list can be broken down into two parts:
  - Its head, i.e., the first element
  - its tail, i.e., the rest of the list after the first element has been removed (this may be empty).
- **Example**: in the list [fred, albert, jim]
  - the head is the element 'fred'
  - the tail is the list [albert, jim]
- When a list is matched to notation in the form [X|Y]
  - X is instantiated to the head
  - Y is instantiated to the tail

Working with Lists in PROLOG

- **Example:** Print out all elements in a given list
  - to print the list [a,b,c] means print ‘a’ and then print the list [b,c]
  - to print the list [b,c] means print ‘b’ and then print the list [c]
  - to print the list [c] means print ‘c’ and then print the list []
  - a function call to print the list [a,b,c] will cause ‘a’ to be printed and then ‘b’ and finally ‘c’

```
print([X|Y]) :-
  write(X), /* write is a function to print out a value*/
  nl, /* nl prints a new line */
  print(Y).
```

What would this program do with the following goal?

```
print ([a,b,c,d,e]).
```