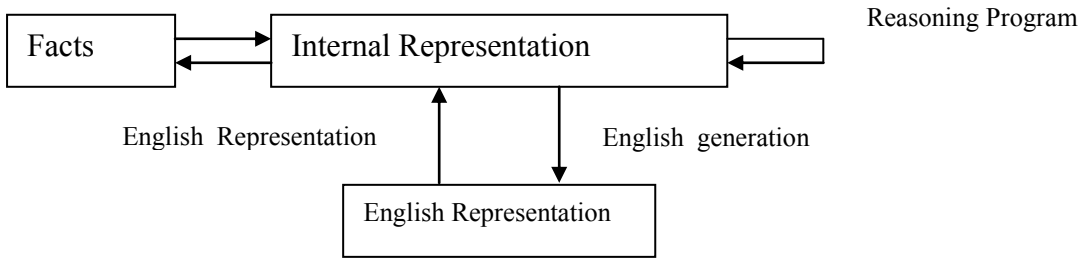
Knowledge & Reasoning
Issues in Knowledge Representation

- Knowledge are facts that can be exploited by AI programs in order to generate good results
- The main entities are:
- Facts: truths in some relevant world
- Representation: the formalism used to describe facts in a way that they can be manipulated.
- These two entities should be structured at two levels:
- Knowledge level – at which facts are described
- Symbol level – at which representation of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs.

Mapping between facts and Representation



Approaches to Knowledge Representation

Properties of good knowledge representation system

- Representational Adequacy – the ability to represent all the kinds of knowledge that are needed in that domain.
- Inferential Adequacy – the ability to manipulate the representational structures in such a way as to derive new structures corresponding to the new knowledge inferred from old.
- Inferential Efficiency – the ability to incorporate into the knowledge structures additional information that can be used to focus the attention of the inference mechanism in the most promising directions.
- Acquisitional Efficiency: - the ability to acquire new information easily. It should be possible to make direct insertion into the database and the addition of new knowledge.

Types of Knowledge Representation

- Simple relational knowledge – using database tables
- Inheritable knowledge – provides for property inheritance in which elements of specific classes inherits attributes and values from more general classes in which they are included. Objects are organized into classes and classes are arranged in a generalization hierarchy. [See Fig. 4.5 page 111: Artificial Intelligence, Elaine, R. and K. Knight]. Examples include Slot-Filler structure like Semantic network and Frames.
- Inferential Knowledge: the power of property inheritance and traditional logic (Predicate, Propositional) are combined to generate the inference (deductions) that is needed.
- Procedural knowledge: facts are not just static or declarative, procedural knowledge specifies what to do and when to do it. This is another kind of knowledge that must be represented. Mostly production rules are used to represent procedural knowledge.
- Semantic knowledge - Ontologies, Vocabulary, Thesaurus, Episodic knowledge

Issues in Knowledge Representation

The following issues cut across the various kinds of real world knowledge.

- Are any attributes of objects so basic that they occur in almost every problem domain? If there are, we need to make sure that they are handled appropriately in each of the mechanisms we propose. If such attributes exist, what are they?
- Are there any important relationships that exist among attributes of objects?
- At what level should knowledge be represented? Is there a good set of primitives into which knowledge can be broken down? Is it helpful to use such primitives?
- How should sets of objects be represented
- Given a large amount of knowledge stored in a database, how can relevant parts be accessed when they are needed?
- (Reference E. Rich, K Knight, 1999, Artificial intelligence)

Answers to Questions

1. (Yes) *instance* and *isa* i.e. class membership and class inclusion are common and support inheritance

2. (Yes)
    a. Inverses: if we can define a relationship from the perspective of objects A and B. It is also possible to define also, an inverse relationship from the perspective of B to A.
    b. Existence in *isa* hierarchy: just as there are classes of and specializations of attributes.

- There exist techniques for Reasoning about values:
    - information about the type of value
    - constraints on the value
    - rules for computing values when it is needed
    - rules that describe action that should be taken, if a value ever becomes known.
- Single-valued attributes
    - explicit notation to track duplication
    - replacement of an old value

Representation should be done at a variety of granularities i.e. both the use of low level primitives and high-level form should be used. But the particular domain will determine which should be more employed.

Exercise

Investigate the frame problem

Knowledge based Agents

- The knowledge base is the central component of a knowledge-based agent.
- A knowledge base is a set of representation of facts about the world, that cause the knowledge based agent to adapt to its environment.
- Each individual representation in a knowledge base is called **Sentence.** The sentences are expressed in a knowledge representation language.
- A Knowledge base must have a **Tell** (assert) and **Ask** (query) mode respectively. In essence knowledge based agent takes a percept as input and returns an action.
- The expectation from a knowledge based agent is that when it asked a question, the response should follow what it has been told which will be the proof of reasoning.
- The knowledge base of a knowledge-based agent is referred to as background knowledge. Every time an agent program is called, two things happens

- 1) it tells the knowledge base what it perceives and
- 2) it asks the knowledge base what action it should perform.

- In doing this, the concept of logical reasoning is used to determine which action is better than others.
- The objective of knowledge representation is to express knowledge in a computer-traceable form, such that it can be used to help agents perform well.
- A knowledge representation is defined by two aspects:
    - The Syntax of a language which describes the possible configuration that can constitute sentences
    - Semantics, which determines the facts in the world to which the sentence refer.
- Provided the syntax and semantics are precisely defined, we can call the language a Logic. Also from the syntax and semantics we can derive an interface mechanism for an agent using the language.
- An Inference procedure can do one of two things:
    - 1. Given a knowledge base KB, it can generate new sentences α that purport to be entailed by KB Or
    - 2. Given a knowledge base KB and another sentence α, it can report whether or not α is entailed by KB. An inference procedure that generalizes sentences is called **sound or truth preserving.**
- Entailment: In mathematical notation, the relation entailment between a knowledge base KB and a sentence α is pronounced "KB entails α" and written as: $KB \models \alpha$.
- An inference procedure i can be described by the sentences it can derive. If i can derive α from KB, this could be written as $KB \models_i \alpha$. i.e. alpha is derived from KB by i or i derives alpha from KB.
- The record of operation of a sound inference procedure is called a **proof.**
- An inference procedure is complete if it can find a proof for any sentence that is entailed.
- In summary:
- Logic consist of:

1. A formal system of describing states of affairs, consisting of:
   a) The syntax of the language, which describes how to make sentences,

b) The semantics of the language, which states the systematic constraints on how sentences relates to states of affairs.

2. The proof Theory- a set of rules for deducing the entailments of a set of sentences.

- Therefore, by examining the semantics of a logical language, we can extract the semantics of the language.

Types of Agent Programs

Simple Reflex Agent

Model-based Agent

Goal-based Agent

Utility-based Agent

Learning Agent

Knowledge Representation with First-Order Logic

- **Propositional logic** can also be used but it lacks sufficient primitives for representing knowledge in a logical way. It has limited ontology, making only the commitment that the world consists of facts. But this is not true and grossly inadequate.
- Predicate logic (Firs-order logic) makes a stronger set of ontological commitments. It has primitives to represent objects, relations properties and functions. The main reason for this is that the world consists of an object, that is, things with individual identities and properties that distinguish them from other objects. Among these objects various relations hold. Some of these are functions-relations in which there is one "value" for a given "input". Examples of objects, properties, relations and functions include:
- Objects: people, houses, numbers, ball etc.
- Relations: brotherof, biggerthan, inside, part of, hascolor etc.
- Properties: red, round, yellow, prime
- Functions: fatherof, bestfriend etc.
- First-order logic has **sentences**, but it also has **terms,** which represents objects. Constant symbols, variables and function symbols are used to build terms, and quantifiers and predicate symbols are used to build sentences.

Syntax of First-Order Logic (with equality in Backus-Nomal Form)

Sentences → Atomic sentences
| sentence connective sentence
| Quantifier variable… Sentence
| ┐ sentence | ┐ (sentence)

Atomic sentence → Predicate (Term, …) | Term = Term

Term → Function (Term,…) | Constant | Variable

Connective → =>| Λ | V | <=>

Quantifier → ∀ | ∃

Constant → A | X_1 | John

Variable → a | x | s…

Predicate → Before | Hascolor | Raining | ---

Function → Mother| Leftlegof |

- **A term** is a logical expression that refers to an object. The formal semantics of terms is that an interpretation must specify which object in the world is referred to by the function symbol and objects referred to the terms of its arguments.
- **Constant symbol**: specifies which object is referred to by each constant symbol. e.g. A,B,ADE etc.
- **Predicate symbol**: refer to a particular relation in the model.
- **Function symbol**: These are functional relation that relates any given object to exactly one other object. E.g. any angle has only one number that is its cosine, any person has only one person as his or her father.
- **Atomic sentences**: is formed from a predicate symbol followed by a parenthesized list of terms. For example:
- Brother(Richard, John)
- This sentence states Richard is the brother of John.
- Atomic sentences can have arguments that are complex terms e.g.
- Married (fatherof (Richard), Motherof (John))- This states that the father of Richard married the mother of John. An atomic sentence is true if the relation referred to by the predicate symbols holds between the objects referred to by the arguments.
- Complex sentences: A complex sentence is combination of two or more atomic sentences using logical connections e.g.
- Brother(Richard, John) Λ Boss(Richard, John) – is true when Richard is the brother of John and Richard is the Boss of John.
- Older(John, 30) => ┐Younger (John, 30) – if John is older than 30, then he is not younger than 30
- **Quantifiers:** Quantifiers helps to express properties of entire collections of objects, rather than having to enumerate objects by name. The two standard quantifiers in First-order logic are called **Universal and Existential operators**
- ∀ cat(x) => mammal(x). This is pronounced for all x, if x is a cat then x is a mammal.
- Thus the universal quantifier makes statements about every object in the universe. The existential quantifier allows us to make statements about some object in the universe without naming it.
- ∃ sister (x, spot) Λ cat (x) – This is pronounced " There exist x, such that x is a sister to spot and x is a cat

Exercise

Represent the following sentences in first-order logic using a consistent vocabulary you must define.

a) Not all student take both History and Biology
b) Only one student failed History
c) Only one student failed both History and Biology
d) The best score in History was better than the best score in Biology
e) Every person who dislikes all vegetarian is smart
f) No person likes smart vegetarians
g) There is a woman who likes all men who are not vegetarians
h) There is a barber who shaves all men in town who do not shave themselves
i) No person likes a Professor unless the professor is smart
j) Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

Sample Solution

a) ∀ student(x) => ┐ (take(x, History) Λ take(x, Biology))

b) ∀ student(x, onlyone) = student_failed(x, history)

c) ∀ student(x, onlyone) => student_failed(x, history) Λ
student_failed(x, biology)

d) ∀ betterthan(bestscore(x, history), bestscore(x, biology))

(e) ∀ person(x) Λ dislike(x, vegetarian) → smart(x)

Semantic Networks

A *semantic network* or *net* is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics.

What is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge. Some versions are highly informal, but other versions are formally defined systems of logic. Following are six of the most common kinds of semantic networks:

1. *Definitional networks* emphasize the *subtype* or *is-a* relation between a concept type and a newly defined subtype. The resulting network, also called a *generalization* or *subsumption* hierarchy, supports the rule of *inheritance* for copying properties defined

for a supertype to all of its subtypes. Since definitions are true by definition, the information in these networks is often assumed to be necessarily true.
2. *Assertional networks* are designed to assert propositions. Unlike definitional networks, the information in an assertional network is assumed to be contingently true, unless it is explicitly marked with a modal operator. Some assertional netwoks have been proposed as models of the *conceptual structures* underlying natural language semantics.
3. *Implicational networks* use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.
4. *Executable networks* include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations.
5. *Learning networks* build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called *weights*, associated with the nodes and arcs.
6. *Hybrid networks* combine two or more of the previous techniques, either in a single network or in separate, but closely interacting networks.

Some of the networks have been explicitly designed to implement hypotheses about human cognitive mechanisms, while others have been designed primarily for computer efficiency.

"Knowledge representation is an issue that arises in both cognitive science and artificial intelligence. In cognitive science it is concerned with how people store and process information. In artificial intelligence (AI) the primary aim is to store knowledge so that programs can process it and achieve the verisimilitude of human intelligence.

In cognitive theory

For some authors knowledge is stored either in episodic or semantic memory. The further is organized in spacio-temporal dimensions, the second according semantic content-oriented principles, e.g. networks of concepts.

"Long-Term Memory which is a large storage system, stores factual information, procedural rules of behavior, experiential knowledge , in fact everything we know. We have two types of long term memory Episodic and semantic memory Episodic memory represents our memory of events and experiences in a serial form. It is from this memory that we can reconstruct the actual events that took place at a given point in our lives. The second is Semantic memory, which is a structured record of facts, concepts and skills that we have acquired. The information in semantic memory is derived from that in our episodic memory, such that we can learn new facts or concepts from our experiences. Semantic memory is structured in some way to allow access to information representation of relationship between pieces of information and inference. One model for the way in which semantic memory is structured is as a network. Items are associated to each other in classes and may inherit attributes from parent classes. This model is known as a Semantic network.

In computer science

In computer science, a semantic network can be defined as a knowledge representation formalism which describes objects and their relationships in terms of a network consisting of labelled arcs and nodes.

- "A semantic network is often used as a form of knowledge representation. It is a directed graph consisting of vertices which represent concepts and edges which represent semantic relations between the concepts."
- "A semantic network is a knowledge representation tool consisting of a framework of semantically related terms, with the purpose of allowing a definition of those words through their relationships."
- Most ontologies use a kind of semantic network for knowledge representation.

The advantages of knowledge representation structure like semantic network over First-Order Logic includes the fact that:

1. It makes it easy to describe properties of relations
2. It is a form of object-oriented programming and has the advantages that such systems normally have, including modularity and ease of viewing by people.

Here is an example of semantic nets:

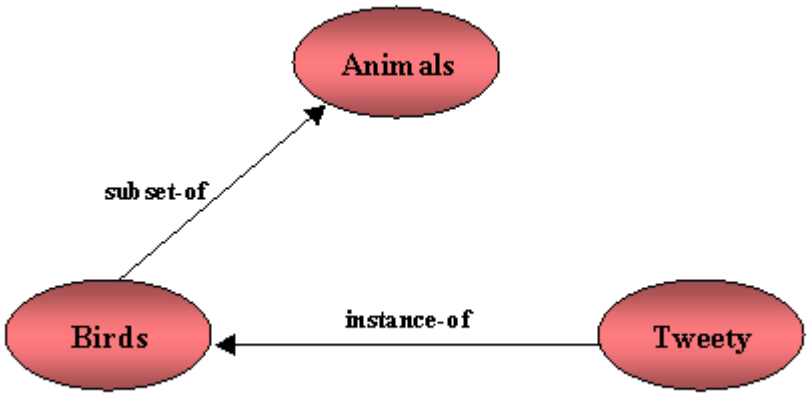


Figure 1. Animals-Birds-Tweety

The major problem with semantic nets is that although the name of this knowledge representation language is semantic nets, there is not, ironically, clear semantics of the various network representations. For the above example, it can be interpreted as the representation of a specific bird named Tweety, or it can be interpreted as a representation of some relationship between Tweety, birds and animals.

Semantic networks can also include other explicit kind of relationships among concept and concept types that adequately represent the semantic relationships among entities. As an example, Figure 2 shows a KL-ONE network that defines the concepts Truck and TrailerTruck as subtypes of Vehicle.

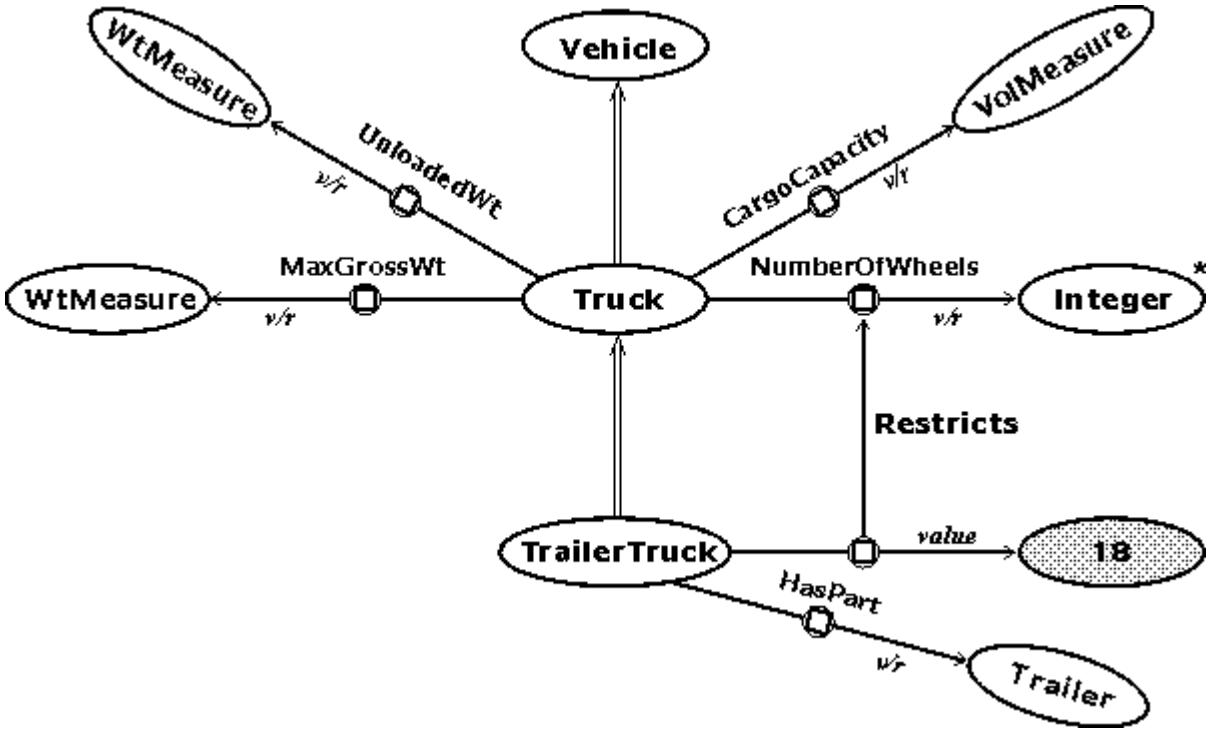


Figure 2: Truck and TrailerTruck in KL-ONE

Figure 2 has nine ovals for concept nodes and nine arrows, which represent different kinds of links. The white ovals represent *generic concepts* for the types, as distinguished from the shaded oval, which is an *individual concept* for the instance 18. The oval marked with an asterisk * indicates that Integer is a built-in or primitive type. The concepts Truck and TrailerTruck are defined in Figure 2, but Vehicle, Trailer, WtMeasure, and VolMeasure would have to be defined by other KL-ONE diagrams.

The double-line arrows represent subtype-supertype links from TrailerTruck to Truck and from Truck to Vehicle. The arrows with a circle in the middle represent *roles*. The Truck node has four roles labeled UnloadedWt, MaxGrossWt, CargoCapacity, and NumberOfWheels. The TrailerTruck node has two roles, one labeled HasPart and one that restricts the NumberOfWheels role of Truck to the value 18. The notation *v/r* at the target end of the role arrows indicates *value restrictions* or type constraints on the permissible values for those roles.

Generally, formal graph notations annotated with specific labels can be used for semantic network representations.

Exercises

Show the representation of the following: i) Student ii) Teacher iii) Worker iv) Computer.

Learning

What is Learning?

- Learning is the process through an entity acquires knowledge.
- Machine intelligence is not natural and automatic, an intelligent machine is one that exhibits intelligence after a process of learning.

Approaches to Learning:

- **Symbolic learning**: describes systems that formulate and modify rules, facts, and relationships, explicitly represented in words or symbols. In other words they create and modify their own knowledge base.
- **Numerical learning**: refers to systems that use numerical models, where certain techniques are used for optimizing the numerical parameters. Examples include neural networks, genetic algorithms and simulated annealing.
- Learning can be with a teacher in which case it is said too be *supervised*. *Unsupervised learning* is learning without a teacher.

Classification of learning

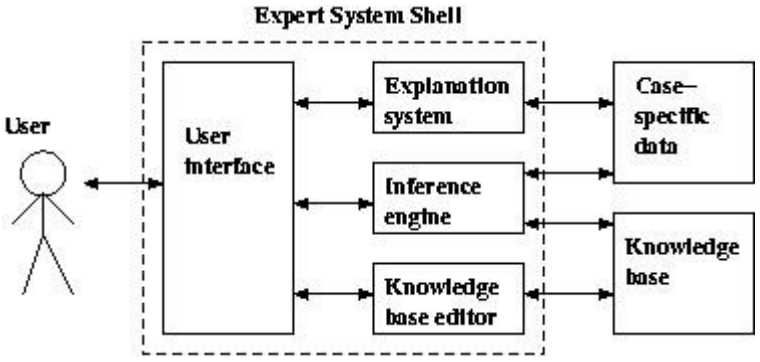
- **Rote Learning**: The system is given confirmation of correct decisions. When it produces incorrect decision it is "spoon fed" with the correct rule or relationship that it should have used.
- **Learning from Advice**: Rather than being given a specific rule that should apply in a given circumstance, the system is given a piece of general advice, such as "gas is more likely to escape from a valve than from a pipe". The system must sort out for itself how to move from this high level advice to an immediately usable rule.
- **Learning by Induction**: The system is presented with sets of example data and is told the correct conclusion that is should draw from each. The system continually refines its rules and relations so as to correctly handle each new example.
- **Learning by Analogy**: The system is told the correct response to a similar, but not identical task. The system must adapt the previous response to generate a new rule applicable to the new circumstances.
- **Explanation-Based Learning (EBL)**: The system analyzes a set of examples solutions and their outcomes to determine why each one was successful or otherwise. Explanations are generated, which are used to guide future problem solving. An example of an EBL system is PRODIGY (a general purpose problem-solver).
- **Case-Based Reasoning (CBR)**: Any case about which the system has reasoned is filed away, together with the outcome, whether it is successful or otherwise. Whenever a new case is encountered, the system adapts its stored behaviour to fit the new circumstances.
- **Explorative or Unsupervised Learning**: This is also called discovery learning, rather than having an explicit goal, an explorative system continuously searches for patterns and relationships in the input data, perhaps marking some patterns as interesting and warranting further investigation. Examples of application of unsupervised learning can be found :
    - data mining : where patterns are sought among large or complex data sets;
    - Identifying clusters, possibly for compressing the data;
    - Feature recognition

2. Expert systems

- Expert systems are meant to solve real problems which normally would require a specialised human expert (such as a doctor or a minerologist).
- Building an expert system entails extracting the relevant knowledge from the human expert and representing the 'heuristic knowledge' in *knowledge base*.
- A *knowledge engineer* has the job of extracting this knowledge and building the expert system. This is called Knowledge Elicitation and Encoding.
- The most widely used knowledge representation scheme for expert systems is rules (sometimes in combination with frame systems).
- Typically, the rules won't have certain conclusions - there will just be some degree of certainty that the conclusion will hold if the conditions hold. Statistical techniques are used to determine these certainties. Rule-based systems, with or without certainties, are generally easily modifiable and make it easy to provide reasonably helpful traces of the system's reasoning. These traces can be used in providing explanations of what it is doing.
- Expert systems have been used to solve a wide range of problems in domains such as medicine, mathematics, engineering, geology, computer science, business, law, defence.

Expert System Architecture

Figure 1 shows the most important modules that make up a rule-based expert system. The user interacts with the system through a *user interface* which may use menus, natural language or any other style of interaction). Then an *inference engine* is used to reason with both the *expert knowledge* (extracted from our friendly expert) and data specific to the particular problem being solved. The expert knowledge will typically be in the form of a set of IF-THEN rules. The *case specific data* includes both data provided by the user and partial conclusions (along with certainty measures) based on this data. In a simple forward chaining rule-based system the case specific data will be the elements in *working memory*.



Almost all expert systems also have an *explanation subsystem*, which allows the program to explain its reasoning to the user. Some systems also have a *knowledge base editor* which help the expert or knowledge engineer to easily update and check the knowledge base. `

One important feature of expert systems is the way they (usually) separate domain specific knowledge from more general purpose reasoning and representation techniques. The general purpose bit (in the dotted box in the figure) is referred to as an *expert system shell*. As we see

in the figure, the shell will provide the inference engine (and knowledge representation scheme), a user interface, an explanation system and sometimes a knowledge base editor. Given a new kind of problem to solve (say, car design), we can usually find a shell that provides the right sort of support for that problem, so all we need to do is provide the expert knowledge. There are numerous commercial expert system shells, each one appropriate for a slightly different range of problems. (Expert systems work in industry includes both writing expert system shells and writing expert systems using shells.) Using shells to write expert systems generally greatly reduces the cost and time of development (compared with writing the expert system from scratch).

Examples : MYCIN, XCON(R1),PROSPECTOR