Knowledge Representation

CSIP5403 – Research Methods and AI Applications

Dr. Nathanael L. Baisa

Lecture Content

Representation Basics:

- Data types (Boolean, integer, float, array, records, lists, etc.)

Knowledge Representation (KR) Structures:

- Decision Trees
- Semantic Networks
- Scripts
- Frames
- Logic
- Rules

Constructing a System: From Representation to a System

Knowledge Acquisition

Building an Expert System

Session Outcomes

- Acquire proper understanding of knowledge representation basics
- Understand Knowledge Representation (KR) structures in detail
- Gain understanding of how to construct a system from representation
- Understand what Knowledge Acquisition (KA) is in detail
- Gain knowledge of how to build an Expert System (ES)

Representation Basics

Information to Knowledge

Many **AI systems** attempt to find answers to questions imposed by **human users**. E.g.

- I have these symptoms, do I have cold?
- From these images, do a patient has cancer?
- From the historical data of the London stock exchange in which of FTSE100 companies, should we invest?
- In a smart home, how can energy use be reduced, possibly based on its inhabitants habits? Are there other factors should or could be taken into consideration in reducing the energy use?
- Similar questions apply to mathematical problems, space exploration, car manufacturing, and many more.

What do they all have in common?

Information to Knowledge

All of the examples we looked at so far and any more you may wish to list would have a common thread of using **information** gathered in structured or unstructured data format.

But **data**, regardless of how well structured it may be, can rarely provide conclusions that can help decision making process or enable finding new knowledge.

Thus, data need to be processed, interpreted, and **knowledge** needs to be extracted for the purpose of driving intelligent systems and meeting application requirements.

Hence the need for **Knowledge Representation**.

Information to Knowledge

All of the examples we looked at so far and any more you may wish to list would have a common thread of using **information** gathered in structured or unstructured data format.

But **data**, regardless of how well structured it may be, can rarely provide conclusions that can help decision making process or enable finding new knowledge.

Thus, data need to be processed, interpreted, and **knowledge** needs to be extracted for the purpose of driving intelligent systems and meeting application requirements.

Hence the need for **Knowledge Representation**.

Knowledge representation and reasoning (KRR) is the study of how to represent information about the world in a form that can be used by a computer system to solve and reason about complex problems.

Knowledge Types

Often knowledge is defined into **two types**:

- **Shallow**: which focuses on surface or apparent knowledge represented often in simple rules with limited or no data use. This is often used in Reactive Systems i.e.AI systems that have no memory and are task specific.
- **Deep**: which focuses on data and the inter-relationships between data units.

However, we shall start our exploration of knowledge representation from its simplest data types that we will need to use in defining our basic variables when modeling.

In a simple system of one sensor, we may only need to record one variable of a Boolean type, such as the case with a switch.

However, we are likely to have sensors that give us variety of values.

For multi-component systems, we may combine data types into structures.

Then, we may combine structures into models or knowledge structures.

Data Types

Data types are usually the first to be covered in any programming language course and form the basis for data structures.

Just as a reminder, data types are:

- Boolean
- Integer
- Float
- Character

How many **bits** or **bytes** are used for each of these data types?

Why is it important to know the size of a data type? (hint: think of limited resource devices such as small mobile robots deployed in environments with limited accessibility).

Array Based Structures

One of the most basic and yet widely used data structures is the **Array**.

Arrays provide the bases for a number of data structures such as **Stacks** and **Queues**.

An array is simply an indexed collection of entries usually sharing the same data type.

Main operations that may be used are:

- Insertion
- Deletion
- Sorting
- Search
- Retrieve

Defining the strategies of inserting and retrieving data from an array enable us to define other data structures such as stack.

Direct access to data within the array is possible using the array index.

Records

In multi-component systems, there are likely to be **multiple data types** that form a single unit of information and thus should be kept together in a record.

The “struct” statement in C allows us to create a data type that encapsulate multiple data units each with its own type.

Classes and database records enable us to do the same.

Lists

There is also often the need to link these structures to each other to express “order”, “causation” or other types of relationships.

Lists provide the data structure in which links between data units can be established.

Arrays can be used for simple lists.

However, we often use Linked Lists leading to a more complex structures or representation such as Trees and other Graph based knowledge representation.

The nodes within linked lists can contain variety of data types/structures including complex records.

Review the implementation of these data structures in your programming language.

KR Structures

KR Structures

There are common structures that became de facto part of knowledge representation. These structures are often based or related to three approaches to organizing knowledge.

- **Graph based** structures such as lists are extended into Decision Trees, Semantic Networks, and cognitive maps, etc.
- **Descriptive structures** such as classes extended to Frames, Scripts, and agents.
- **Logic based structures** such as most of programming languages statements extended to Rules which are the cornerstone of Rule-based Systems.

Usually we start this topic by covering logic, but instead we will cover examples that we can connect together, so from Lists that we have just covered to one of the most important KR structures in use: Trees.

Decision Trees

Lists can be organized in a tree by having each node with left and right pointers (branches).

A tree is a key data structure that embeds hierarchical and relational information.

Useful way of modelling knowledge for search algorithms (e.g. depth first, breadth first, best first, A*, etc.) and hence for expert systems.

In particular, decision trees can be useful for modelling rule-based systems. They can also be derived from existing data – inductive learning.

Disadvantage: no clear definition of the relationships between the nodes.

If we can define set of relationship types between nodes we can encode semantic information, so comes in Semantic Networks.

Semantic Networks

Consists of nodes and links like lists and trees.

Show inheritance.

Can override (e.g. penguin doesn't fly), this is something logic-based representations often struggle with.

No standard link names, hence a bit 'messy'.

Semantic Networks

Advantage: Easy to translate a semantic network to a PROLOG database.

```
ako (chair, furniture)
ako (chair, seat)
isa (your_chair, chair)
isa (you, person)
made_of (your_chair, wood) colour (wood, brown) belongs_to (your_chair, you)
colour (wood, brown)
belongs_to (your_chair, you)
```

NB: ako – a-kind-of, isa – is-a.

Disadvantage:

- Can't embed heuristic knowledge.
- Represent shallow knowledge.

Scripts

There is evidence that humans organize knowledge into structures corresponding to typical situations.

Scripts were designed by Schank, 1977, as organizing conceptual dependency structures into descriptions of typical situations.

Used in natural language understanding.

Can be used in Case-Based Reasoning and in analysis/design phase of system development.

Often used in developing multimedia systems as they enable story telling.

Scripts

Components of scripts:

- **Entry condition**: generally must be satisfied before events if a script can occur.
- **Result**: generally true after events in scripts have occurred.
- **Props**: slots representing objects involved in events within a script.
- **Roles**: slots representing people involved in events within a script.
- **Track**: specific variation of general context represented by the script.
- **Scenes**: actual sequence of events that occur.

Frames

Frames are similar to scripts. Minsky (1975) discusses this as a way that our memories are organised.

Again used to store stereotyped situations.

Contributed to the development of the concepts “Class” and “Object”.

Can look like a network organisation but there are important differences: (has attributes/slots, supports class inheritance, can have procedural attachments, etc.).

Frames

Data structures – containing detailed knowledge about particular object.

Hierarchical structure – usually represent stereotyped knowledge based on well known experiences, etc.

Contain SLOTS – the values etc. associated with the slots are called FACETS.

- SLOT – attributes that describe the object.
- FACET – describes knowledge/ procedures about the attribute. So Facets can be values, default values, ranges of values, etc.

More organized than semantic networks for example.

Uses defaults (corresponds to experience type knowledge).

Logic

There are **two** main types of logic systems traditionally used (but by far not the only ones): **Propositional** and **Predicate**.

Predicate Logic is an extension of **propositional logic**. It enhances representation by allowing the use of variables and functions in addition to the basic logic operations (and, or, not, implies and equivalent), e.g.

Colour (Ball, Red); Likes (John, Mary)

- Colour and Likes are predicates
- Ball, Red, John, Mary are constants

It can also use variables, e.g. Likes(X,Y)

It can express rules scope over variables using **quantifiers**: universal (or for-all) ∀ and existential (or exists) ∃

e.g. $\forall x \forall y$: father(x,y) $\vee$ mother (x,y) $\longrightarrow$ parent (x,y)

This reads: 'For all values of x and y, if x is the father of y, or x is the mother of y, then x is the parent of y'

What does this logic sentence read as?

$\forall x \forall y \forall z$: parent (x,y) $\wedge$ parent (x,z) $\longrightarrow$ sibling (y,z)

Logic

Given the following logical rule:

$$\forall x \forall y: \text{father}(x,y) \vee \text{mother}(x,y) \longrightarrow \text{parent}(x,y)$$

And the following assertions (facts):

father (jeff, kim)
father(jeff, alan)

We can derive:

parent (jeff, kim)

parent (jeff, alan)

Other forms in logic are in use especially Modal, Temporal and Descriptive Logics.

Rules

Rules are a simplified form of logical statements. Premise and consequences are represented together in much simpler format than propositions, predicates and axioms (axiom is a statement or proposition accepted as true).

Each rule will have the form:

IF Premise THEN conclusion

It can be easily combined with other KR structures.

In finding an answer to a question, a search space can be easily constructed from rules.

Disadvantage: good at representing shallow knowledge but struggles at representing complex inter-related knowledge.

As rules may contradict each other, a process is required to manage this process often referred to as **Inference Engine**.

Constructing a System: From Representation to a System

Constructing a System

Knowledge representation enables us to capture information in a format that can be used in constructing a system that enables a user to interact with that information, interrogate it, reason about it and draw conclusions from it.

One of the simplest and most successful AI systems to do this relies on the use of rules and hence called Rule-based Systems.

This type of systems gave rise to the field of Expert Systems and Knowledge-based Systems.

Constructing a system around represented knowledge is important for developing solutions using AI algorithms for different applications.

From Presentation to a System

Imagine you have a well organized data in a database, how useful is this would be without any queries that enable you to interrogate the data?

Perhaps an SQL search query would help?

A system that enables us to navigate the data and answer questions? Like an expert?

Having Knowledge representation on its own is not enough so we need to build a **system** that allows us to use this representation. However, the first step in building such a system is to gain the domain knowledge that we are trying to represent and utilize. This is a keystone step in Expert and Knowledge-Based Systems called **Knowledge Acquisition**.

Knowledge Acquisition

What is KA?

Knowledge Acquisition (KA) is the process of acquiring knowledge, which also implies:

- Collecting information from multiple sources within a given domain.
- Organizing the information into format that could be analyzed.
- Analyzing the information to extract knowledge.

KA is usually considered to be a bottleneck in Expert Systems (ES) development.

Why? Take 5 minutes to discuss the issues surrounding the three activities of KA that could cause difficulties in completing the process.

Sources of Knowledge

In acquiring knowledge there are usually **two main types** of sources of knowledge:

- **Documented**: this usually comes in many forms, such as company policies, patient's forms, records, data collections, etc.
- **Undocumented**: this is usually unique, important and most difficult to acquire because it is often maintained in the mind of the expert, e.g. think of a doctor's surgical skills.

What are the difficulties of acquiring knowledge? Discuss these difficulties for each source.

Categories of Knowledge

We have already seen the two types of knowledge: Shallow and Deep.

The knowledge acquired can be categorised into **three** main categories:

- **Declarative** – which declares facts (i.e. descriptive knowledge).
- **Procedural** – which provides procedures of operation and the use of declarative knowledge.
- **Semantics** – which provides relationships between concepts (e.g. words and symbols), their meaning, and how they are related and manipulated. This often attempts to reflect cognitive structure.

What form of representation would be most suitable for each category? As examples:

- **Declarative** – logical propositions
- **Procedural** – rules
- **Semantics** – semantic networks

Techniques

As KA is a difficult task, there are several techniques and methods that have been developed to capture knowledge from its different forms. A **Knowledge Engineer (KE)** is likely to use a combination of these techniques since a single technique is unlikely to be sufficient. Examples of these techniques are:

- **Interviews** – there are several techniques that could be used:
  - Expert focused interview
  - Structured interview
  - 'Thinking aloud' interview
- **Protocol analysis** during which domain expert is observed performing a task while explaining actions. This is usually recorded.
- **Observation** – a more generic form of protocol analysis, it can save experts time; however, it could be time consuming for KE and may require clarifications.
- **Case studies** – examining specific cases either that may be available from variety of sources, e.g. from published literature, public records, or provided by experts.
- **Automated** – using specialized software tools which often focus on rule extraction.

Building an Expert System (ES)

ES Components

Expert System usually consists of three main components:

Knowledge Base, which contains the representation of domain knowledge.

Working memory, which maintains the current system state and where updates of the known facts are added. This is usually the result of the inference mechanism cycles.

Inference Engine, which enables the use of this domain knowledge to answer the user questions.

Knowledge Base

Knowledge base contains the knowledge captured during the knowledge acquisition process in an appropriate representation.

Rules "IF…THEN" proved to be most successful representation to be used, hence ES sometimes referred to as Rule-based Systems. However, other representations have been used in recent years prompting the use of Knowledge-based Systems. Why?

Rules suffer from limited knowledge representation capabilities where certainty, probability and vagueness are difficult to be represented to name but few.

Fuzzy rules are used to develop Fuzzy Systems variation of ES.

Certainty factors are often used to express confidence in conclusions made from processing (firing) rules.

More complex representation may be needed for more complex cases where units of knowledge are inter-linked.

Knowledge Base

Case-Based Reasoning (CBR) uses a library of past cases rather than rules. CBR systems usually utilize the following process:

- Retrieve most similar case or cases to the given problem or query.
- Reuse retrieved case(s) to try to solve current problem.
- Revise and adapt proposed solution(s) if necessary.
- Retain final solution as part of a new case.

How does this relate to knowledge representation techniques:

- Scripts?
- Frames?

Working Memory

Working memory can be seen as the notepad of ES where initial facts about the system state are asserted to initiate the system.

Then after each cycle of rule evaluation and processing, the working memory is updated by:

- Deleting declarative statements (Facts) that are no longer true.
- Asserting new facts that are the conclusion of firing or evaluating rules.

Working memory is usually implemented as an array.

As knowledge representation became more complex in ES, working memory structure and implementation became varied with some complex models often inspired by psychological and neuro-psychological studies of **human memory**.

How can we use Artificial Neural Networks (ANNs) as a memory?

Inference Engine

Inference engine is the heart of ES and its primary function is to evaluate the rules.

Rule evaluation is done through search algorithms, where search space is constructed from all available or possible facts, i.e. the antecedent and/or the consequent parts of the rules.

Various search algorithms can be used to enable rule evaluation, however, traditional ES have usually used two main strategies in rule evaluation: forward chaining and backward chaining.

Rule Inference Strategies

Forward Chaining: starts with known facts (model of current system state) and reason or infer forward with the given rules.

- Test each rule and fire every rule whose antecedent (IF part) can be shown to be true.
- Assert conclusion(s) in working memory.
- Test the rules again, which continues iteratively until no more rules are fired.

Backward Chaining: starts from conclusion or goal and attempts to find the route to achieve it.

- Test each rule that has the goal in the consequent (THEN part).
- Search antecedents to see what we need to know to fire that rule. This leads to sub-goals.
- Repeat testing the rules that can lead to proving the sub-goal. The process of stacking the rule and creating more sub-goals continues Until:
  - The rule under examination can be fired and the goal can be proved.
- Once this point is reached, we can evaluate the last rule, un-stack it and proceed to the next, etc.

Other Inferencing Mechanisms

Using other knowledge representation techniques would require its own inference mechanisms.

Similarly, advanced AI algorithms are used in developing inference engines that are better capable of dealing with more complex knowledge and its inter-links.

A good example is ANFIS (Adaptive Neuro-Fuzzy Inference System), which was developed by Jang (1993) doi:10.1109/21.256541

ANFIS uses neural network approach to develop an inference engine to evaluate fuzzy rules. It is an example of an ever growing family of Hybrid Systems.

Genetic Algorithms (GA) can be used to select ANFIS parameters to optimize the system further.

Conclusion

- Knowledge representation is very crucial for the development of intelligent systems e.g. rule-based systems, expert systems, etc.
- In acquiring knowledge, there are usually two main types of sources of knowledge: documented and undocumented.
- Expert System usually consists of three main components: knowledge base, working memory and inference engine.
- Traditional expert systems have usually used two main strategies in rule evaluation or inference: forward chaining and backward chaining.

References

S. Russel, P. Norvig, ‘Artificial Intelligence: A Modern Approach’, Pearson Series, 2021. (available online).

R.J. Brachman and H. J. Levesque, ‘Knowledge Representation & Reasoning’, Elsevier, 2004. (available online).

K. Darlington, The Essence of Expert Systems, Prentice-Hall, Essex, England (2000).