AITA : Knowledge Representation

1. What is Knowledge?
2. What is a Knowledge Representation?
3. Requirements of a Knowledge Representation
4. Knowledge Representation in Natural Language
5. Databases as a Knowledge Representation
6. First Order Logic as a Representation
7. Rule Based Systems
8. Semantic Networks
9. Frame Based Systems
10. Which Knowledge Representation is Best?

What is Knowledge?

The Chambers 20th Century Dictionary provides as good a definition as any:

> **knowledge**, *nol´ij*, n. assured belief; that which is known; information; …

In order to solve the complex problems encountered in AI, one generally needs a large amount of knowledge, and suitable mechanisms for representing and manipulating all that knowledge.

Knowledge can take many forms. Some simple examples are:

John has an umbrella
It is raining
An umbrella stops you getting wet when it's raining
An umbrella will only stop you getting wet if it is used properly
Umbrellas are not so useful when it is very windy

So, how should an AI agent store and manipulate knowledge like this?

What is a Knowledge Representation?

The object of a ***knowledge representation*** is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

A ***knowledge representation language*** is defined by two aspects:

1. **Syntax** The syntax of a language defines which configurations of the components of the language constitute valid sentences.
2. **Semantics** The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

This is a very general idea, and not restricted to natural language.

Suppose the language is arithmetic, then

‘*x*’, ‘≥’ and ‘*y*’ are *components* (or symbols or words) of the language

the *syntax* says that ‘$x \geq y$’ is a valid sentence in the language, but ‘$\geq \geq$ x y’ is not

the *semantics* say that ‘$x \geq y$’ is false if *y* is bigger than *x*, and true otherwise

Requirements of a Knowledge Representation

A good knowledge representation system for any particular domain should possess the following properties:

1. **Representational Adequacy** – the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. **Inferential Adequacy** – the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.
3. **Inferential Efficiency** – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. **Acquisitional Efficiency** – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a 'knowledge engineer' would be acceptable.

Finding a system that optimises these for all possible domains is not going to be feasible.

Knowledge Representation in Natural Language

Humans usually use ***natural language*** (English, Spanish, Chinese, etc.) to represent knowledge, so why not use that to represent knowledge in our AI systems?

Advantages of Natural Language

1. It is extremely expressive – we can express virtually everything in natural language (real world situations, pictures, symbols, ideas, emotions, reasoning, …).
2. Most humans use it most of the time as their knowledge representation of choice (how many text books are not written in natural language?).

Disadvantages

1. Both the syntax and semantics are very complex and not fully understood.
2. There is little uniformity in the structure of sentences.
3. It is often ambiguous – in fact, it is *usually* ambiguous.

Database Systems

Simple ***databases*** are commonly used to good effect in Computer Science. They can be use to store and manipulate virtually any kind of information.

For example, the database may consist of a number of simple records stored in ASCII format:

```
Person record = {  name : max 20 characters
                   age : 3 digits in range 000-120
                   sex : male or female
                   marital status : single, engaged, married, divorced, widowed
                   children's names : up to 10 names each max 15 characters
                   employer : company code of 3 characters
                 }
```

Generally, we can have any number of fields, containing whatever information we need, in any format.

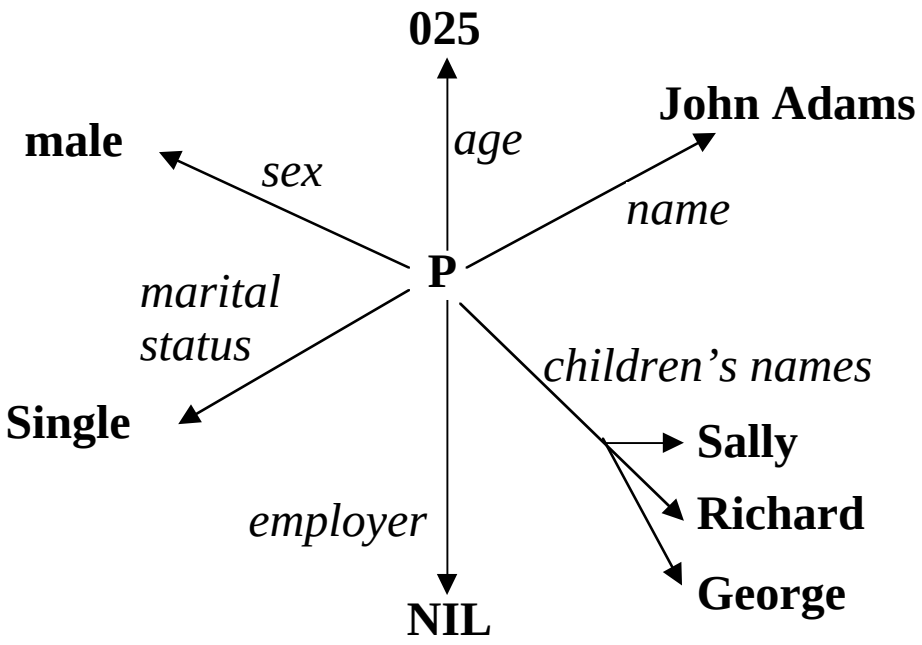
Instances in a Database System

Information in a database can be displayed in a variety of ways, for example:

A Record Structure

John Adams
025
male
single
Sally
Richard
George
NIL

A Directed Graph



But storage and display are not enough – we also need to manipulate the knowledge.

Manipulations of a Database System

We can construct sentences in an appropriate language, for example:

“marital_status(John Adams) is single”	CORRECT
“marital_status(John Adams) is divorced”	INCORRECT SEMANTICS
“marital_status(025) is male”	INCORRECT SYNTAX

We can also generate relations, for example:

R1: Employment

⋮	⋮
NIL	John Adams
NIL	Fred Smith
NIL	Sam Patel
NTL	Jo McNeal
⋮	⋮

R2: Parent/Child

⋮	⋮
John Adams	Sally
John Adams	Richard
John Adams	George
Karen Adams	Susan
⋮	⋮

Databases as a Knowledge Representation

Traditional database systems are clearly very powerful, but for AI systems they are rather limited. In particular:

Disadvantages

1. Only simple aspects of the problem domain can be accommodated.
2. We can represent *entities*, and *relationships* between entities, but not much more.
3. Reasoning is very simple – basically the only reasoning possible is simple lookup.

Advantages

1. Databases are well suited to efficiently representing and processing large amounts of data (and derivation from a database is virtually independent of its size).
2. We can build on traditional database systems to process more complex and more powerful representational devices (e.g. frames).

First Order Logic

The syntax and semantics of ***first order logic*** will be covered in detail next semester.

Some typical sentences in first order logic are:

1. man(William) $\vee$ woman(Karen)
2. married(William, Karen)
3. $\forall x \exists y$[person(x) $\Rightarrow$ has_mother(x,y)]
4. $\forall x \forall y$[[parents(x,y) $\wedge$ man(x)] $\Rightarrow \neg$man(y)

The language consists of constants {William, Karen, etc.}, variables {x, y, etc.}, functions/predicates {Married(x,y), person(x), etc.}, and the logic symbols:

Logic	$\vee$	$\wedge$	$\Rightarrow$	$\neg$	$\forall$	$\exists$
Nat. Lang.	or	and	implies	not	for all	there exists

We can also manipulate the logic representations to generate new knowledge.

First Order Logic as a Knowledge Representation

We can combine sentences by the 'rules of logic' to produce new sentences, e.g.

$$\frac{\neg man(Chris) \qquad \neg man(x) \Rightarrow woman(x)}{woman(Chris)}$$

As a knowledge representation, first order logic has pros and cons:

Advantages

1. It is very expressive.
2. It has unambiguous syntax and semantics.

Disadvantage

1. There is no generally efficient procedure for processing knowledge.

Rule Based Systems

A ***rule based system*** consists of:

1. A ***rule set*** for representing the knowledge structure.
2. A ***database management system*** for the domain specific facts.
3. A ***rule interpreter*** for the problem solving.

A typical rule set might be:

R1. IF Raining $\wedge$ Outside(x) $\wedge$ HasUmbrella(x) THEN UseUmbrella(x)

R2. IF Raining $\wedge$ Outside(x) $\wedge$ $\neg$HasUmbrella(x) THEN GetWet(x)

R3. IF GetsWet(x) THEN CatchCold(x)

R4. IF Sunny $\wedge$ Outside(x) THEN GetSunBurnt(x)

It should be easy enough to set up an appropriate database management system.

Rule based Inference

If we have a knowledge base consisting of ***facts*** and ***rules***, and a rule interpreter to match the rule conditions against the facts, and a means for extracting the rules, then we can derive new knowledge. For example, using the above rule set:

Suppose we have initial facts: Raining, Outside(John) (and we can assume that if John had an umbrella then we would know about it, so ¬HasUmbrella(John))

Then only R2 with 'x = John' matches the facts, hence we can infer GetsWet(John), and so now we have three facts: Raining, Outside(John), GetsWet(John)

Then R3 with 'x = John' matches the facts, so we can also infer CatchesCold(John), and end up with facts: Raining, Outside(John), GetsWet(John), CatchesCold(John)

Note that there is no way we can end up with GetsSunTan(John).

The process of deriving new facts from given facts is called ***inference*****.**

Rule Based Systems as a Knowledge Representation

We can see that rule based systems have many of the properties required of a knowledge representation. However, as always, there are pros and cons:

Advantages

1. These systems are very expressive.
2. The rules lead to a degree of modularity.
3. We can easily introduce procedures for handling certainty factors, and this leads to the possibility of probabilistic reasoning.

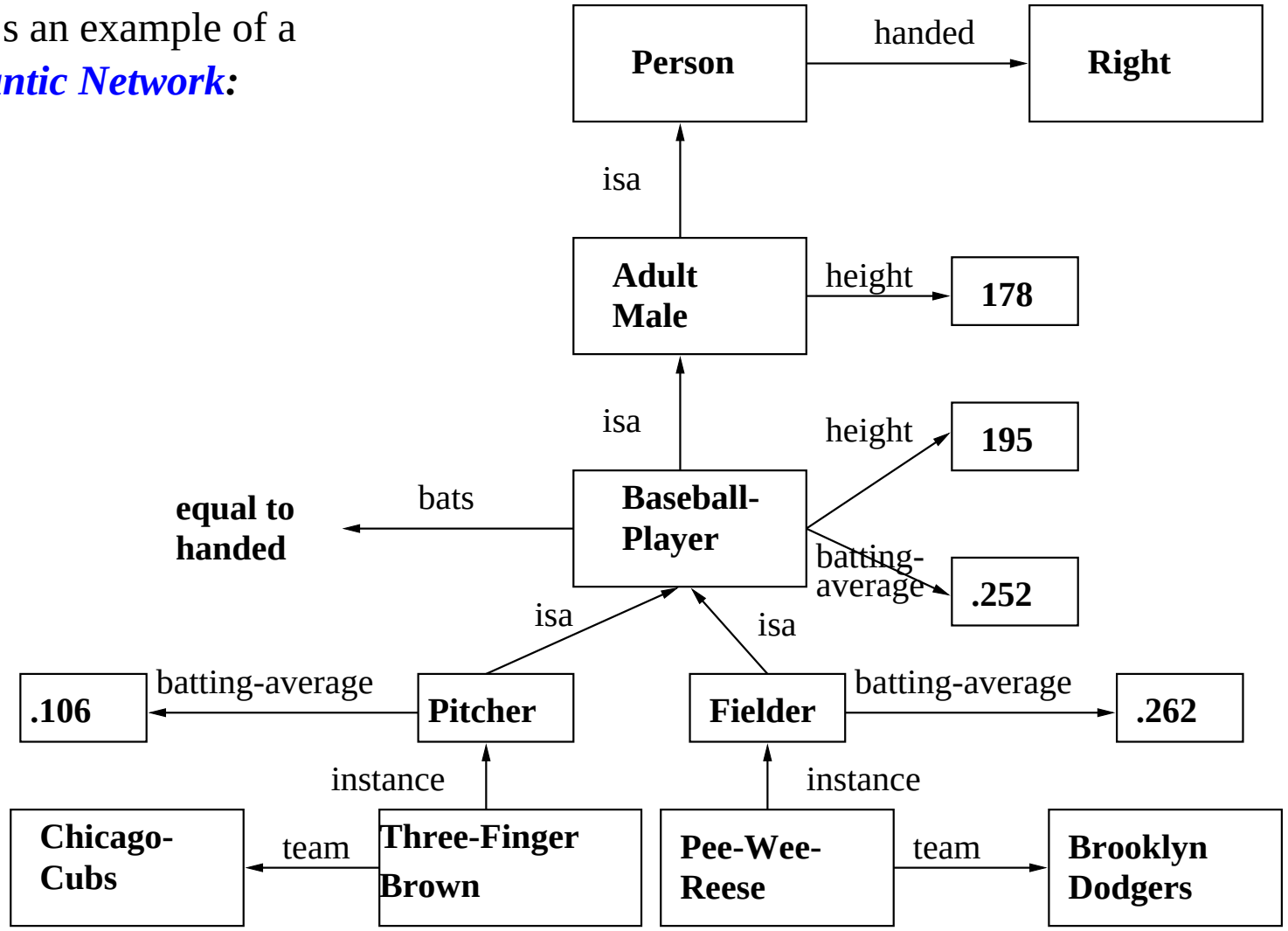
Disadvantage

1. There is a lack of precise semantics for the rules.
2. The systems are not always efficient.

We shall study rule based systems in detail in a series of lectures later in this module.

Semantic Networks

Here's an example of a ***Semantic Network*:**



Semantic Networks as Knowledge Representations

Using Semantic Networks for representing knowledge has particular advantages:

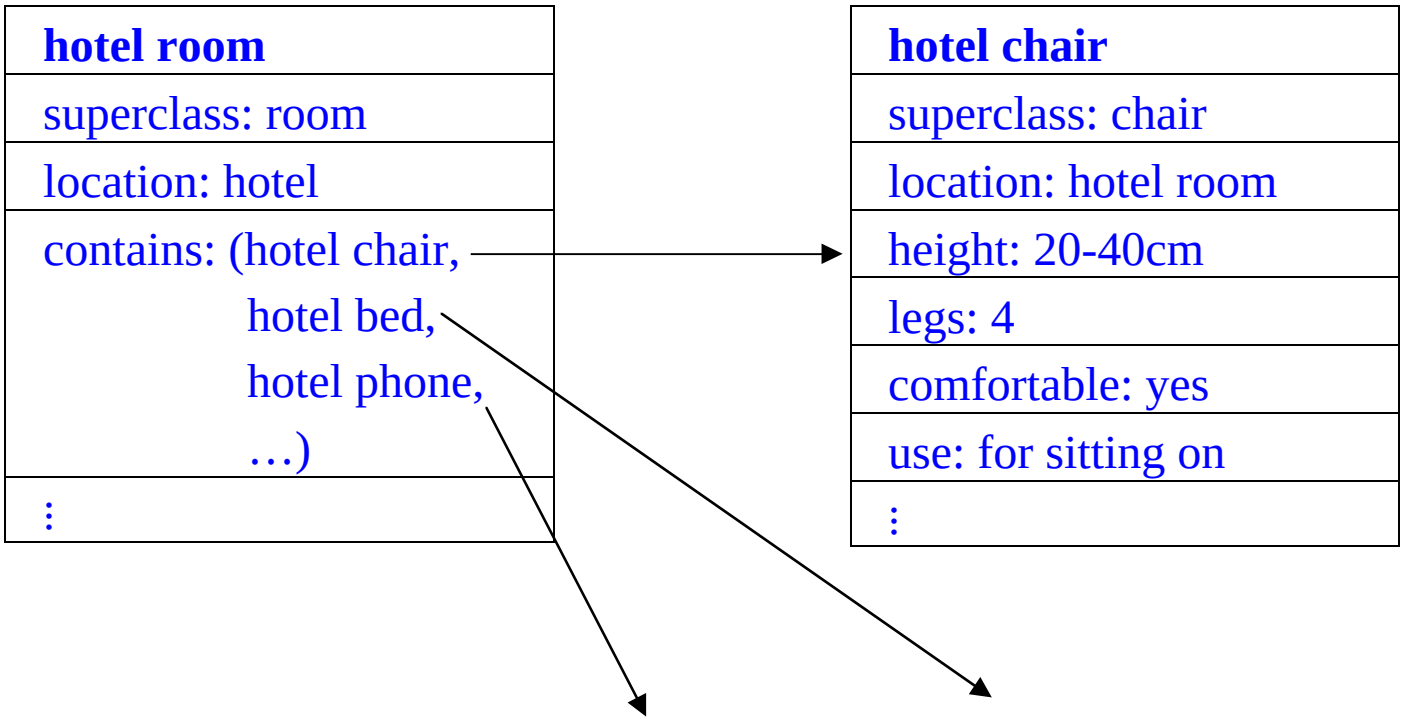
1. They allow us to structure the knowledge to reflect the structure of that part of the world which is being represented.
2. They have a hierarchy of default values (for example, we can assume the height of an adult male to be 178cm, but if we know he is a baseball player we should take it to be 195cm).
3. There are very powerful representational possibilities as a result of “is a” and “is a part of” inheritance hierarchies.

However, the notion of a semantic network is extremely general, and that can actually be a disadvantage. For it to be a useful tool, it must be much refined, and a clear syntax and semantics need to be worked out.

We shall dedicate a whole lecture to Semantic Networks later in this module.

Frame Based Systems

A ***Frame*** consists of a selection of slots which can be filled by values, or procedures for calculating values, or pointers to other frames. For example:



Generally there will be a whole hierarchy of frames to represent the required domain.

Frames as a Knowledge Representation

A frame can simply be a data structure that has similar properties and possibilities for knowledge representation as a semantic network.

Slots in such a frame can also contain instructions (procedures) for computing things from information in other slots or in other frames.

A frame can also be used as a data-structure for representing stereotyped situations, such as going into a hotel room or restaurant. Attached to each frame will then be several kinds of information. Some information can be about how to use the frame. Some can be about what one can expect to happen next, or what one should do next. Some can be about what to do if our expectations are not confirmed. Then, when one encounters a new situation, one can select from memory an appropriate frame and this can be adapted to fit reality by changing the details as necessary.

Clearly we have another very general scheme that needs to be more formalized to be useful. We shall have a whole lecture on Frames later in this module.

Which Knowledge Representation is Best?

We have just taken a very brief tour through a number of the most obviously plausible styles of knowledge representation.

There are clearly many more representational formalisms that might be useful. For a start, we have only really considered ***symbolic*** representations. There also exist ***non-symbolic*** (e.g. pictorial) representations. So-called ***sub-symbolic*** representations are also possible (e.g. as one finds in the activation patterns of neural network systems).

In selecting a representational formalism one needs to consider exactly what has to be represented, and how that knowledge needs to be processed. We should also consider how the system will go about acquiring new knowledge.

The is no single *best knowledge representation* that can be used for everything. In building large complex AI systems, one will usually want to employ many different types of knowledge representation.

Overview and Reading

1. We began by defining what is knowledge, and what is a knowledge representation.
2. We then considered what should be required of a knowledge representation.
3. Then we looked at the possibility of using Natural Language, Databases, First Order Logic, Rule Based Systems, Semantic Networks and Frames as knowledge representations.
4. We concluded that there was no single best type of knowledge representation for all systems, and that large AI systems would probably need to employ more than one type.

Reading

1. Rich & Knight: Chapter 4
2. Russell & Norvig: Chapter 10