Knowledge Representation and Engineering

- Knowledge of Prolog as a KR formalism is assumed, as is AI2.
- Cover more advanced KR material, as well as topics from Knowledge Engineering.
- Should help towards AI4 material on ontologies and agents in the context of the "semantic web" (Multi-agent semantic web systems).

Main Topics

- Symbolic representations and associated meanings
- Logic as a representation language, and deduction as inference
- Efficiency of representation in terms of time and space.
- Reason maintenance systems
- Distributed constraints as distributed reasoning
- Knowledge acquisition methodologies
- Modal logics & associated decision procedures
- Distributed multi-agent architectures

Today

we cover:

- Course organisation
- Provisional Definition of Intelligence
- Knowledge Representation Hypotheses

Course notes will be distributed

Course Organisation

- Lectures by Alan Smaill
- Usual lectures
- Tutorials from week 2
- Two practicals

Tutorial time: Monday 1610

Practicals

There will be 2 practicals –
weeks 3–5;
weeks 6–8.

Course Books

- Russell & Norvig from AI2 is still very useful
- R. Fagin *et al*: "Reasoning about Knowledge", MIT Press, 1995
  Lots of background theory

Provisional definition of intelligence

> Intelligence consists of the principled manipulation of representations, in order to achieve some goal.
> (Aaron Sloman)

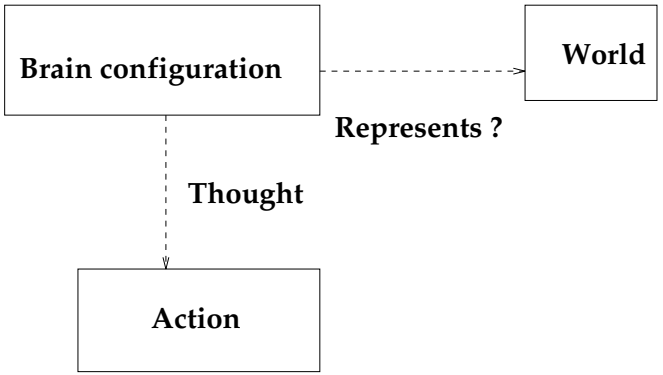
Is this a definition of intelligence in general, and not just AI?

Notice that according to this view, given some *goal*, it should be achieved by some *manipulations* (ie computations ?) applied to some *representation*.

The idea of representation has a mental component —
"X represents Y *for* Z".

Where are the representations?

In natural intelligence, we get a picture as follows:



Symbolic AI and *Connectionism* have different notions of representation.

Suppose an AI program solves some reasoning task. What counts as a *representation* here?

- if program is declarative ?
- if program is compiled from a declarative language?
- if it is a binary with an unknown history?
- if it runs on distributed hardware?
- if it's a connectionist program?

Meaning (semantics)

An old philosophical problem of the nature of *meaning* is raised. New ideas have appeared, based on (eg)

- abstract machines
- compilation
- computer languages

Recall the *computational metaphor*:

> Mind is to body as software is to hardware.

Knowledge Representation Hypothesis

Due to Brian Smith (from his Ph.D. work).

Claim:

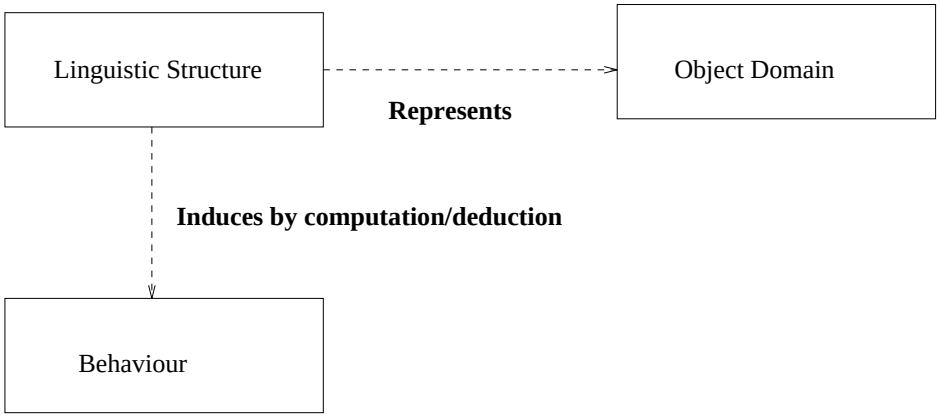
> intelligent processes *must* have component structures that
>
> 1. have a *linguistic* component which
> 2. contains the knowledge of the system, and
> 3. which drives the intelligent behaviour of the system.

KR Hypothesis

– verbatim, this is:

> *Any mechanically embodied intelligent process will be comprised of structural ingredients that*
>
> **a)** *we as observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and*
> **b)** *independent of such external semantic attributes, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.*

This gives the following picture.



Relating natural and artificial representation

How does this relate our previous picture?

We can regard this as more *abstract*:
ignore some of the detail, and also potentially applies to a wider class of situations.

A programming example

```
% Program 1

printColour(hearts) :- !, write('They're maroon').
printColour(hibs)   :- !, write('They're green').
printColour(X)      :- write('Beats me').

% Program 2

printColour(X) :- colour(X,Y),!,
                  write('They're '),write(Y).
printColour(X) :- write('Beats me').
colour(hearts,maroon).
colour(hibs,green).
```


Sorts of knowledge

Recall procedural vs declarative knowledge.

Apart from possessing knowledge (beliefs), humans know *what to do* with the knowledge:

1. how to write arguments

2. how to solve puzzles

Control knowledge is harder to formalise that is declarative knowledge.

eg:

> In a logic program, it's better to put unit clauses before non-unit clauses.

This can be treated non-declaratively, eg via clause ordering, use of non-logical notations (eg cut).

However, this can be thought of declaratively if we use the Reflection Hypothesis.

Reflection Hypothesis

> If we can build a system to reason about the world by manipulating representations, then we can build a system to reason about itself, using a representation of itself.

So, procedural control knowledge can become declarative in the reflective system.

Reflection Hypothesis

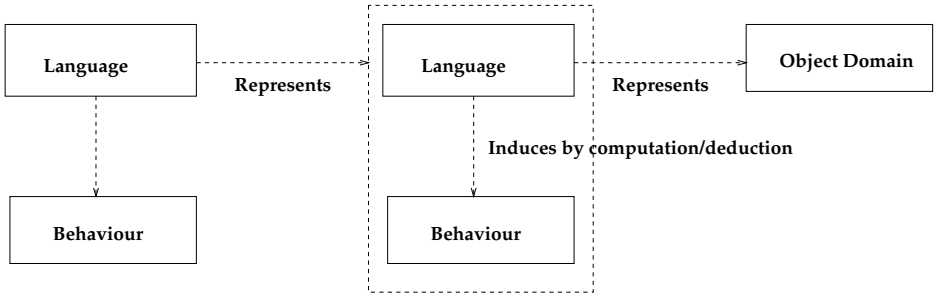
– verbatim, this is:

> *Inasmuch as a computational process can be constructed to reason about an external world in virtue of comprising an ingredient process (interpreter) formally manipulating representations of the world, so too a computational process could be made to reason about itself in virtue of comprising an ingredient process (interpreter) formally manipulating representations of its* own *operations and structures.*

Here, we can think of an agent that can reason about some properties of *itself*.

Reflection Hypothesis

This gives:



Logic as a representation language

A logic plays two roles:

- *Representation* (semantics):
  describes the state of the world
- *Inference* (deduction):
  computable operations that are defines on the syntactic form of the representations.

We will be interested in both of these aspects – and also in the computational *efficiency* of different representations with their associated algorithms.

Summary

- Reasoning as manipulation of representations.
- Knowledge Representation Hypothesis
- Reflection Hypothesis