

Knowledge Representation

COMP9414: Artificial Intelligence

How many rabbits are there?



How many rabbits are there?

- Perception isn't all in the eye.
- Knowledge is usually needed to understand the world



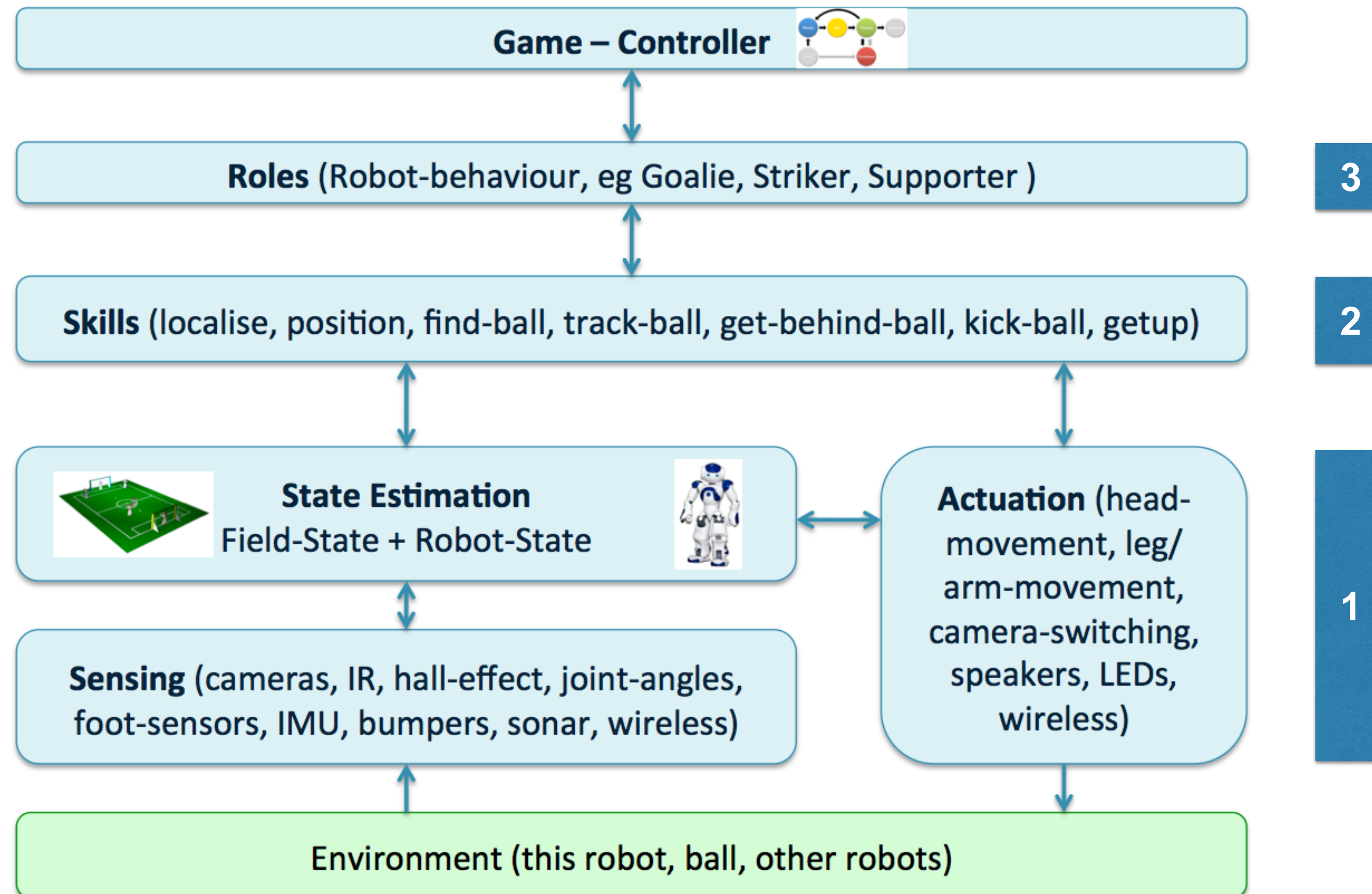
Lecture Overview

- Feature-based vs iconic representations
- Rule-based systems
- Ontological engineering
- Propositional and first-order logic

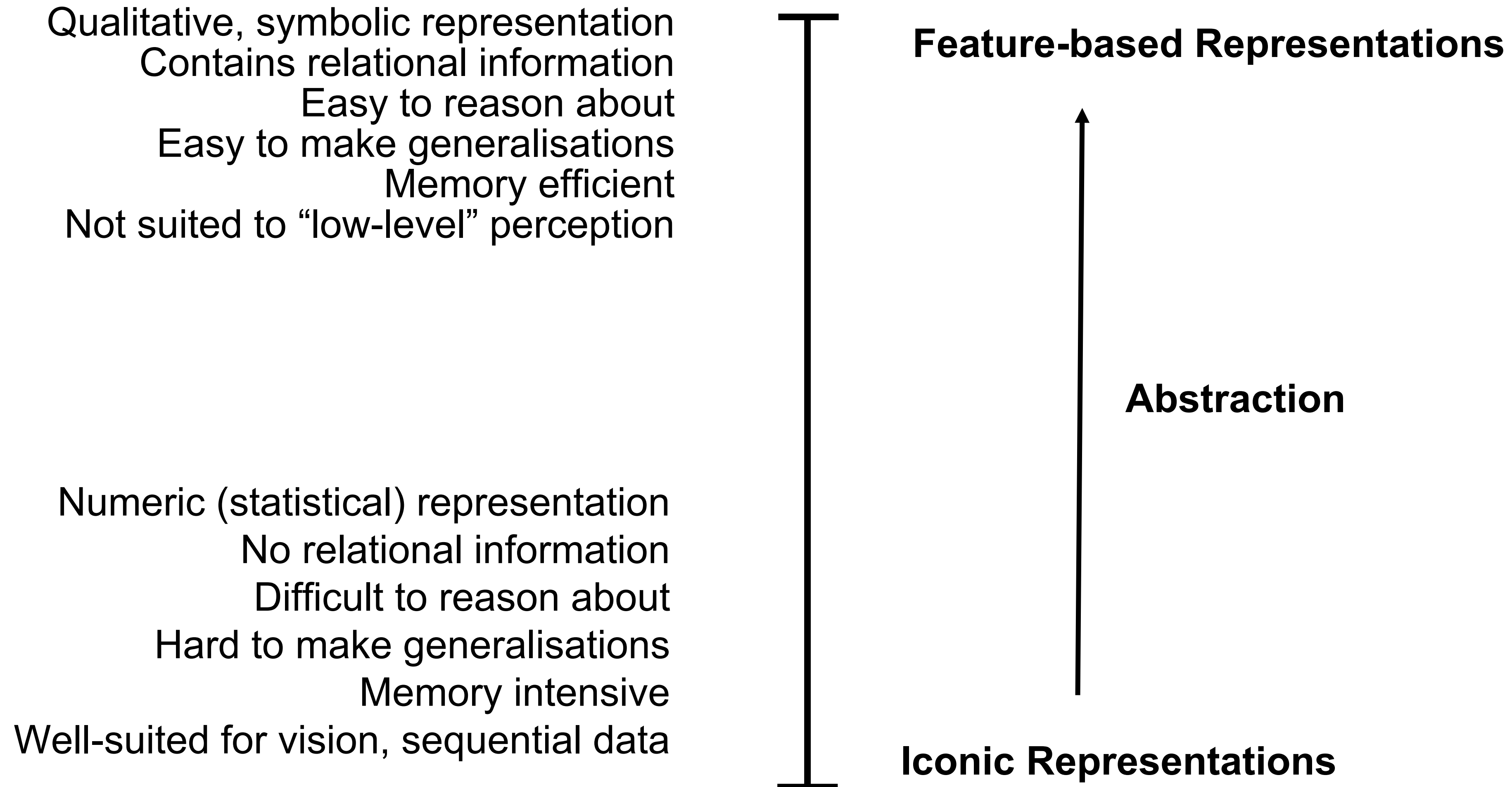
Lecture Overview

- Feature-based vs iconic representations
- Rule-based systems
- Ontological engineering
- Propositional and first-order logic

RoboCup Architecture



Levels of Abstraction



Iconic Representations

- Analogues to real world
 - Pixel representations like first layer of ANN
 - Maps
- Memory-based (requires a lot of memory)
- Fast, but ...
 - Do not generalise well
 - Difficult to perform inferences beyond pattern-response

Symbolic Representations

- State is represented by a set of abstract features and relations
 - For instance: expressions on logic, entity-relation graphs
- Can do complex reasoning over knowledge base
- Not well-suited to "low-level" perception

Knowledge Representation and Reasoning

- A knowledge-based agent has at its core a **knowledge** base
- A knowledge base is an explicit set of **sentences** about some domain expressed in a suitable formal representation language
- Sentences express facts (**true**) or non-facts (**false**)
- Fundamental questions
 - How do we write down knowledge about a domain/problem?
 - How do we automate reasoning to deduce new facts or ensure consistency of a knowledge base?

Lecture Overview

- Feature-based vs iconic representations
- Rule-based systems
- Ontological engineering
- Propositional and first-order logic

Rule-Based Systems

- A **production rule** has the form

```
if <condition> then <conclusion>
```

- Production rule for dealing with the payroll of ACME, Inc., might be

```
rule r1_1
```

```
if the employer of Person is acme
```

```
then the salary of Person becomes large.
```


Rule-Based Systems

```
rule r1_1  
  
if the employer of Person is acme  
  
then the salary of Person becomes large.
```

- Production rules can often be written to closely resemble natural language

```
/* fact f1_1 */  
  
the employer of joe_bloggs is acme.
```

- Capitalisation (like Prolog) indicates that “Person” is a variable that can be replaced by a constant, such as “joe_bloggs” or “mary_smith”, through pattern matching.

Rule-Based Systems

rule r1_1

if the employer of Person is acme

then the salary of Person becomes large.

rule r1_2

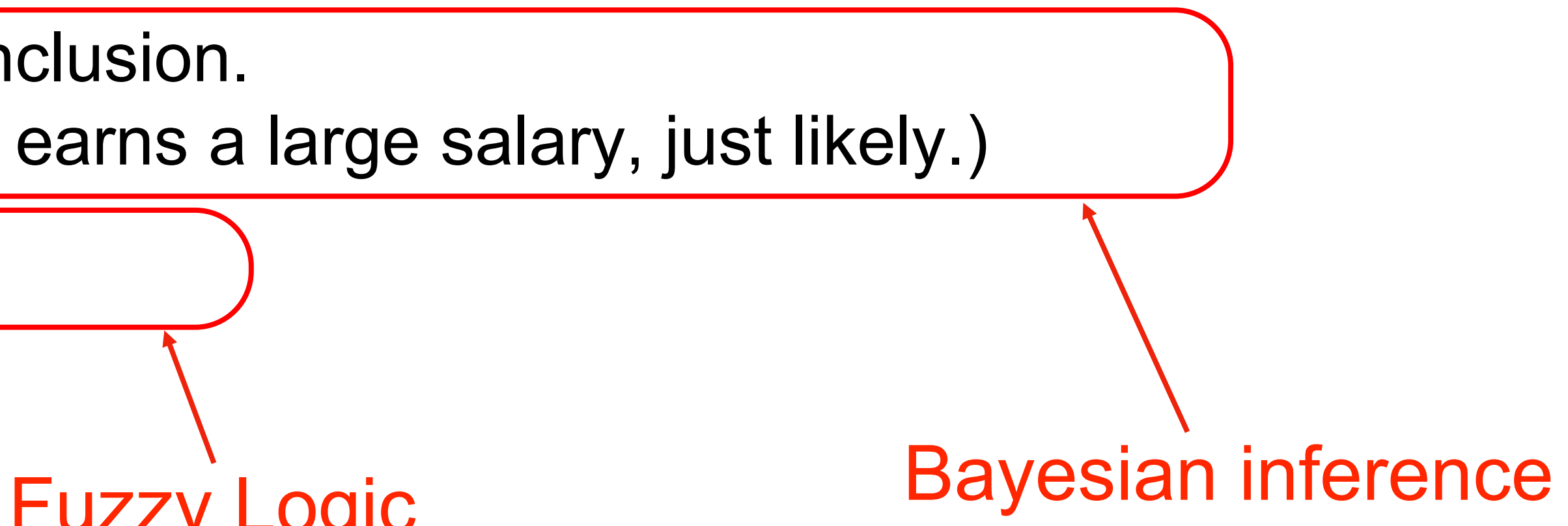
if the salary of Person is large

or the job_satisfaction of Person is true

then the professional_contentment of Person becomes true.

- Executing a rule may generate a new derived fact.
- There is a *dependency* between rules r1_1 and r1_2 since the conclusion of one can satisfies the condition of the other.

Uncertainty in rules

- Rules can express many types of knowledge
 - But how can *uncertainty* be handled?
 - Uncertainty may arise from:
 - Uncertain evidence (Not certain that Joe Bloggs works for ACME.)
 - Uncertain link between evidence and conclusion.
(Cannot be certain that ACME employee earns a large salary, just likely.)
 - Vague rule. (What is a “large”?)
- Fuzzy Logic
- Bayesian inference
- 

Rule-Based Systems

```
rule r1_1
```

```
if the employer of Person is acme
```

```
then the salary of Person becomes large.
```

```
rule r1_2
```

```
if the salary of Person is large
```

```
or the job_satisfaction of Person is true
```

```
then the professional_contentment of Person becomes true.
```

```
/* fact f1_1 */
```

```
the employer of joe_bloggs is acme.
```

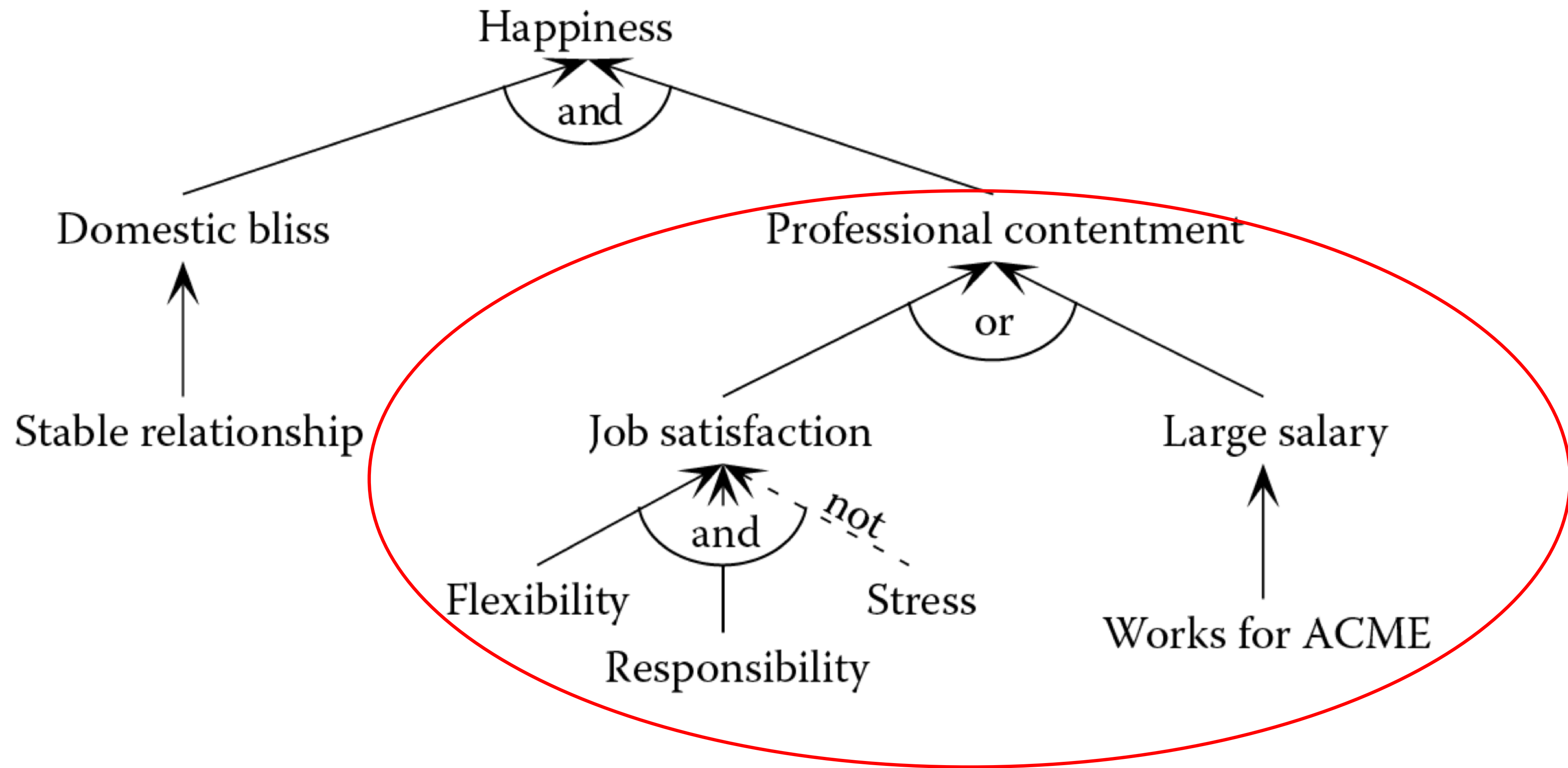
```
/* derived fact f1_2 */
```

```
the salary of joe_bloggs is large.
```


Inference Networks

- The interdependencies among rules, such as r1_1 and r1_2 define **a network**
- **Inference network** shows which facts can be logically combined to form new facts or conclusions
- The facts can be combined using “**and**”, “**or**” and “**not**”.
 - Professional contentment is true if either job satisfaction or large salary is true (or both are true).
 - Job satisfaction is achieved through flexibility, responsibility, and the absence of stress.

An Inference Network



Professional contentment is true if either job satisfaction or large salary is true (or both are true).

Deduction, Abduction and Induction

Rules that make up inference network can be used to link cause and effect:

if <cause> then <effect>

For example:

if

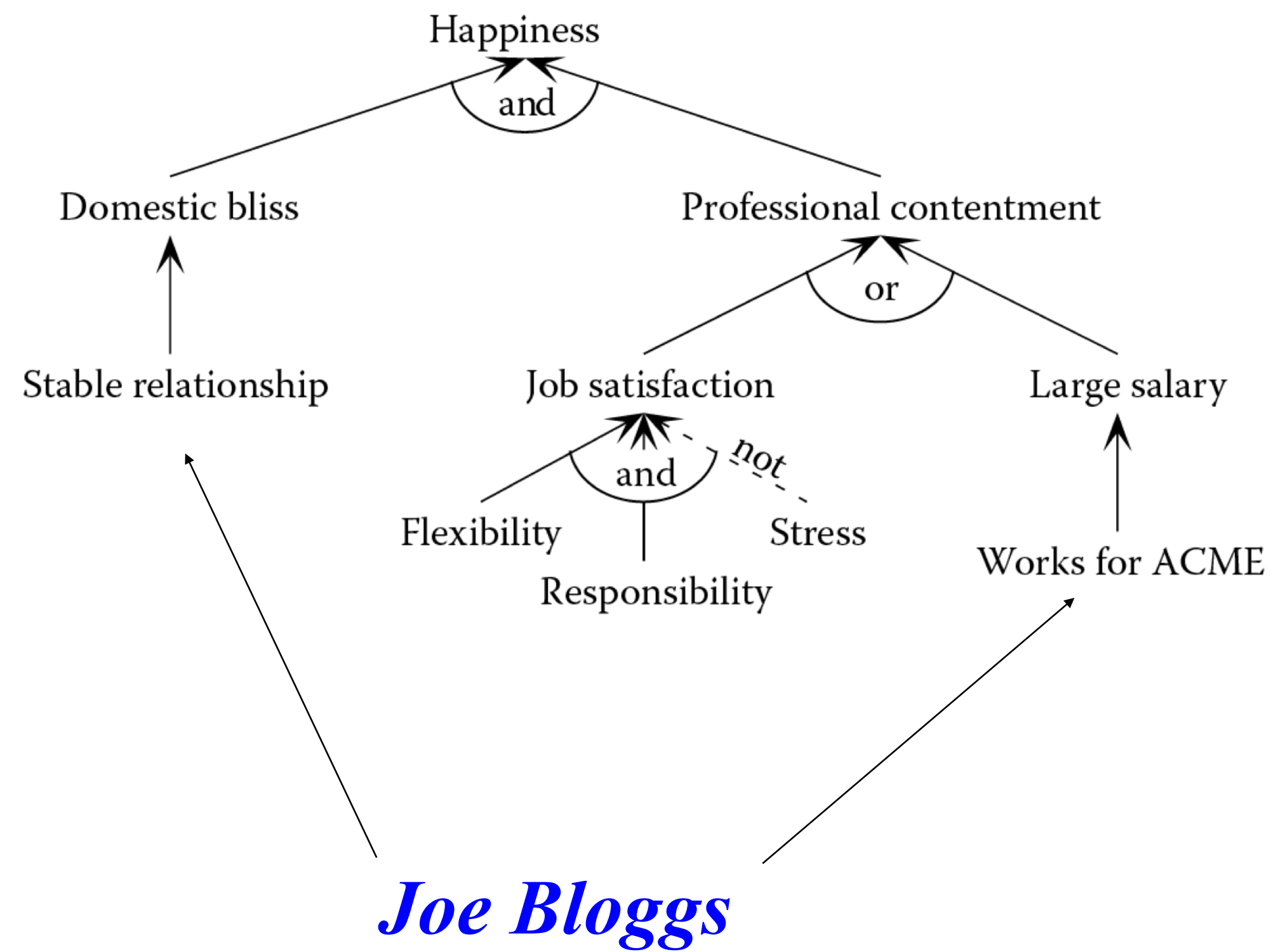
Joe Bloggs works for ACME and is in a stable relationship (the causes),

then

he is happy (the effect).

Deduction, Abduction and Induction

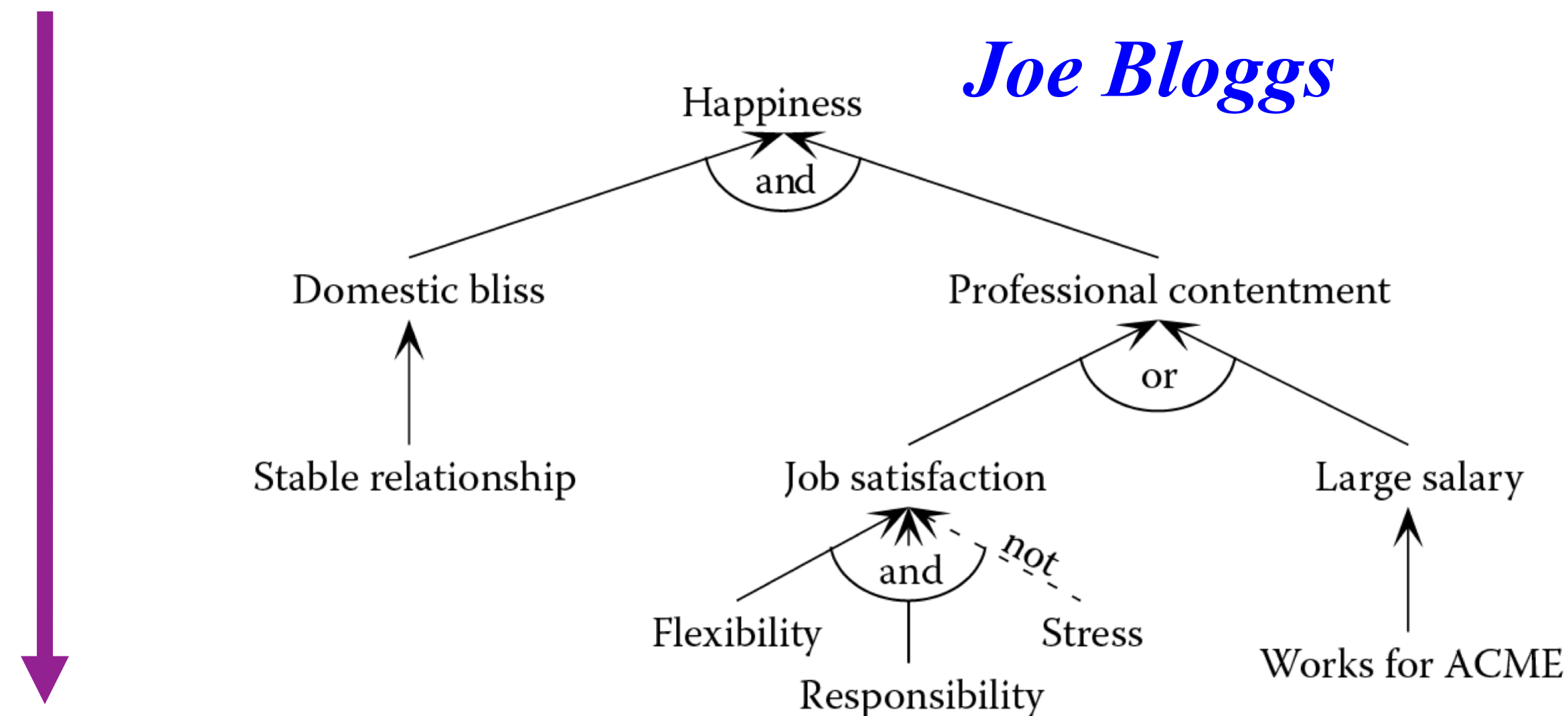
if <cause> then <effect>



if *Joe Bloggs* works for ACME and is in a stable relationship (**causes**), then he is happy (**effect**).

Deduction, **Abduction** and Induction

- Abduction - Many problems, such as diagnosis, involve reasoning in reverse, i.e, find a **cause**, given **an effect**.
- Given observation **Joe Bloggs is happy**, infer by abduction **Joe Bloggs enjoys domestic bliss and professional contentment**.



Deduction, Abduction and Induction

- If we have many examples of cause and effect, infer the **rule** that links them.
- For instance: if every employee of ACME earns a large salary, infer:

rule r1_1

if the employer of Person is acme

then the salary of Person becomes large.

- Inferring a rule from a set of examples of cause and effect is **induction**.

Deduction, Abduction and Induction

- deduction: $\text{cause} + \text{rule} \Rightarrow \text{effect}$
- abduction: $\text{effect} + \text{rule} \Rightarrow \text{cause}$
- induction: $\text{cause} + \text{effect} \Rightarrow \text{rule}$

Closed-World Assumption

- Only facts that are in the knowledge base or that can be derived from rules are assumed to be true
- Everything else is false, i.e., if we don't know it, it's assumed to be false.
- That's why it's more accurate to say:
 - “a proof fails”, instead of “it's false”
 - “a proof succeeds” instead of, “it's true”

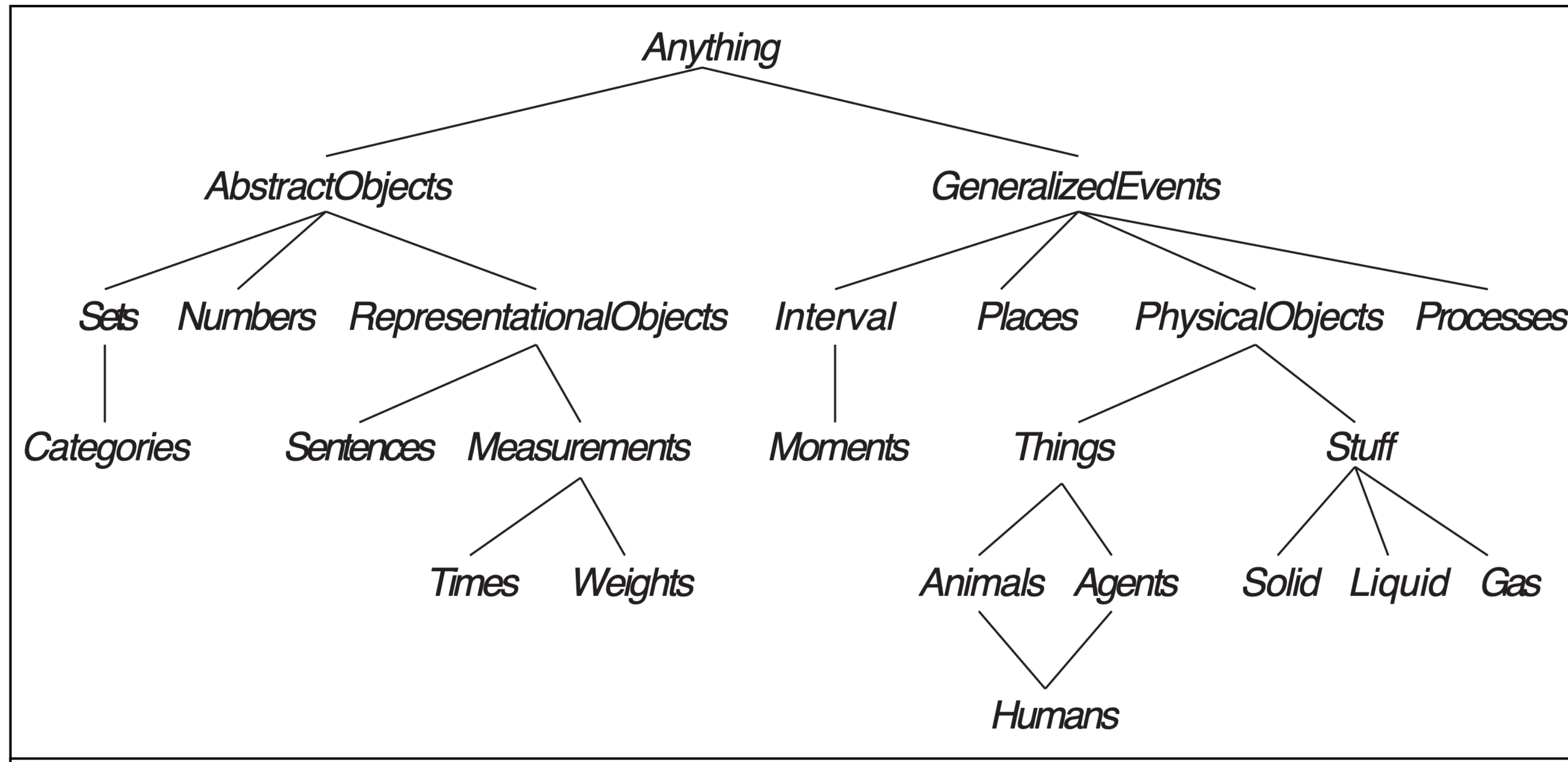
Lecture Overview

- Feature-based vs iconic representations
- Rule-based systems
- **Ontological engineering**
- Propositional and first-order logic

Ontologies and Ontology Engineering

- An **ontology** organises everything into a hierarchy of categories.
- Can't actually write a complete description of everything
 - far too much
 - can leave placeholders where new knowledge can fit in.
 - e.g., define what it means to be a physical object
 - details of different types of objects (robots, televisions, books, ...) filled in later
- Similar to object oriented (OO) programming framework

Ontology Example



- Child concept is a specialisation of parent
- Specialisations are not necessarily disjoint (a human is both an animal and an agent)

Categories and Objects

- Organising objects into categories is vital for knowledge representation.
- Interaction with world takes place at level of individual objects, but ...
 - much reasoning takes place at level of categories
- Categories help make predictions about objects once they are classified

Categories and Objects

- Infer presence of **objects** from perceptual input
- Infer category membership from perceived properties of the objects
- Use category information to make predictions about the objects
- For instance: green and yellow mottled skin & 30cm diameter & ovoid shape & red flesh, black seeds & presence in the fruit aisle → watermelon

Categories and Objects

- **Categories** organise and simplify knowledge base through **inheritance**.
 - if all instances of category Food are edible, and
 - if Fruit is a subclass of Food and Apples is a subclass of Fruit, then
 - infer that every apple is edible.
- Individual apples **inherit** property of edibility
 - in this case from membership in the Food category.

Taxonomic hierarchy

- Subclass relations organise categories into a [taxonomy](#), or [taxonomic hierarchy](#)
- Taxonomies have been used explicitly for centuries in technical fields.
 - Taxonomy of living things organises about 10 million living and extinct species
 - Library science has developed a taxonomy of all fields of knowledge
- Taxonomies are also an important aspect of general [commonsense knowledge](#)

Reasoning system for categories

- Categories are the building blocks of knowledge representation schemes
- Two closely related families of systems:
 - semantic networks:
 - graphical aids for visualizing a knowledge base
 - efficient algorithms for inferring properties of object based in category membership
 - description logics:
 - formal language for constructing and combining category definitions
 - efficient algorithms for deciding subset and superset relationships between categories

Semantic Networks

- Facts, Objects, Attributes and Relationships
 - Relationships exist among instances of objects and classes of objects.
- Attributes and relationships can be represented as a network, known as an **associative network** or **semantic network**
- We can build a model of the subject area of interest

Example – A simple set of statements

- My car is a car
- A car is a vehicle
- A car has four wheels
- A car's speed is 0 mph
- My car is red
- My car is in my garage
- My garage is a garage
- A garage is a building
- My garage is made from brick
- My car is in High Street
- High Street is a street
- A street is a road

Underline = object (instance)

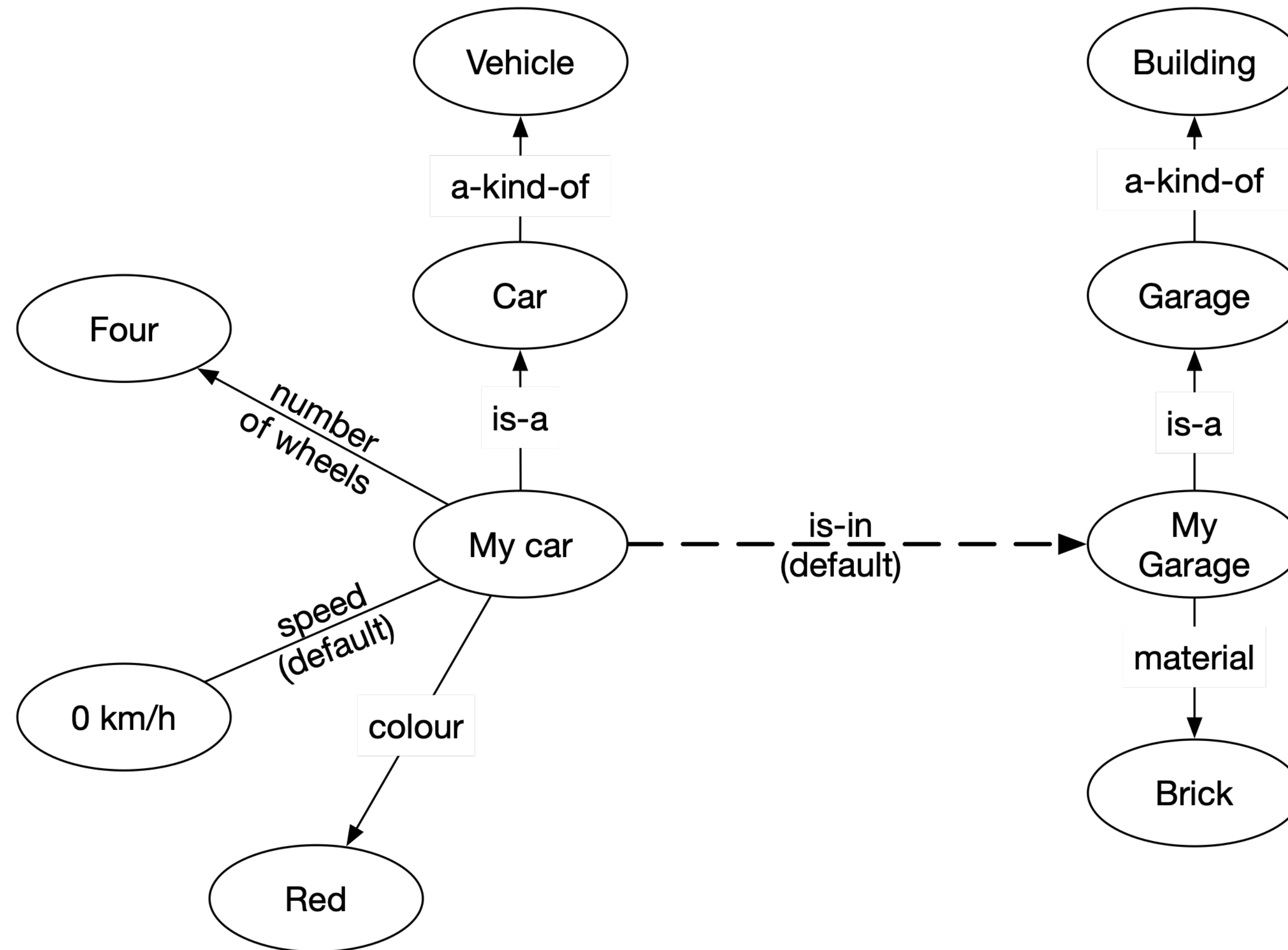
Example – facts, objects and relations

- My car **is a** car
- A car **is a** vehicle
- A car **has four wheels**
- A car's **speed is 0 mph**
- My car **is red**
- My car **is in my garage**
- My garage **is a** garage
- A garage **is a** building
- My garage **is made from brick**
- My car **is in High Street**
- High Street **is a** street
- A street **is a** road

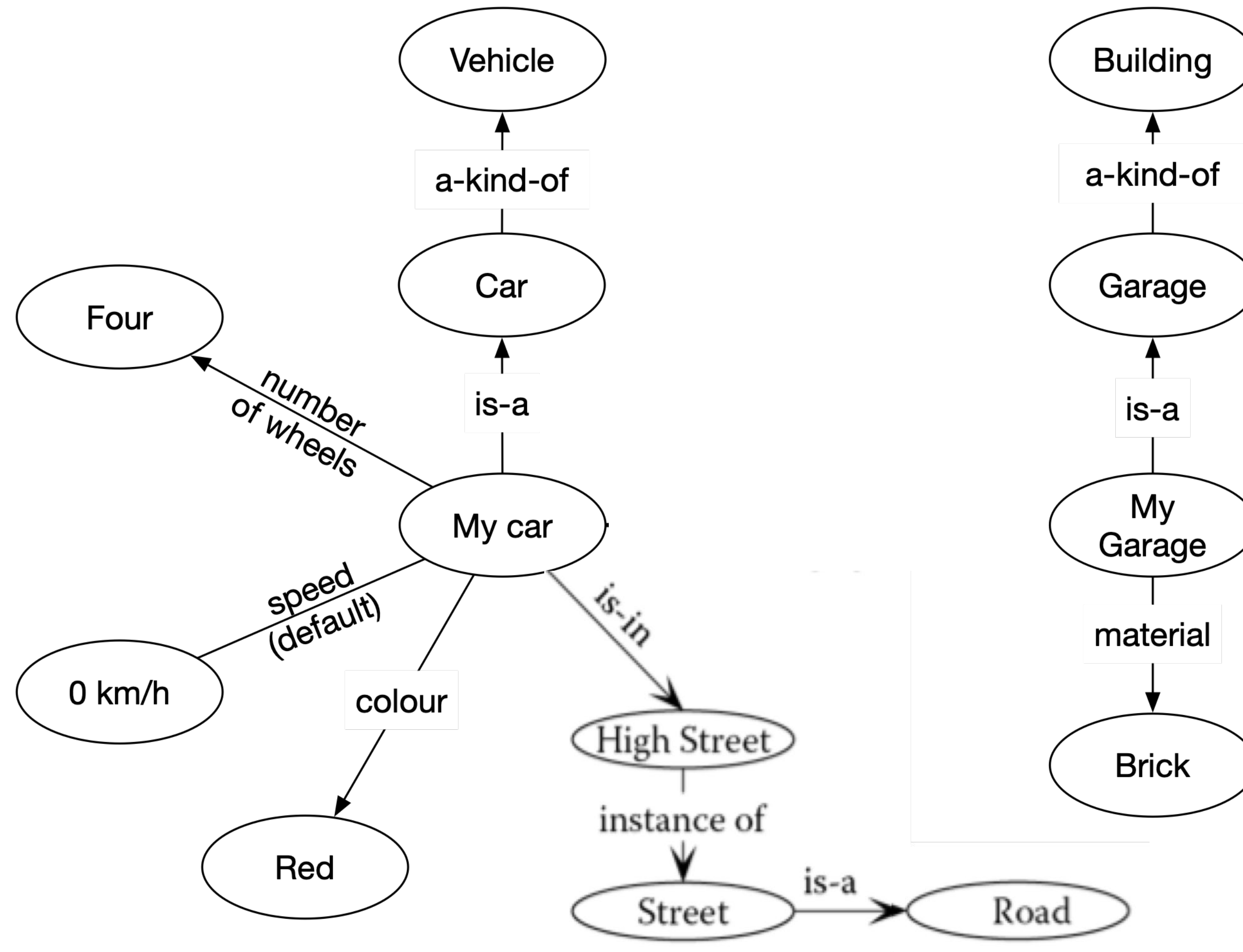
Example – facts, objects and relations

- My car **is a** car (static relationship)
- A car **is a** vehicle (static relationship)
- A car **has four wheels** (static attribute)
- A car's **speed is 0 mph** (default attribute)
- My car **is red** (static attribute)
- My car **is in my garage** (default relationship)
- My garage **is a** garage (static relationship)
- A garage **is a** building (static relationship)
- My garage **is made from brick** (static attribute)
- My car **is in High Street** (transient relationship)
- High Street **is a** street (static relationship)
- A street **is a** road (static relationship)

A semantic network (with a default)



A semantic network (with a default)



Classes and Instances

- Distinction between object instances and classes of objects:
 - Car and vehicle are both classes of objects
 - Linked by “ako” relation (a-kind-of)
 - Direction of arrow indicates “car is a vehicle” and not “vehicle is a car”
- *My car* is a unique entity.
 - Relationship between *my car* and *car* is “isa” (is an instance of)

Lecture Overview

- Feature-based vs iconic representations
- Rule-based systems
- Ontological engineering
- Propositional and first-order logic

Knowledge Bases

- A knowledge base is a set of sentences in a formal language.
- Declarative approach to building an agent:
 - Tell the system what it needs to know,
then it can ask itself what it needs to do
 - Answers should follow from the knowledge based.
- How do you formally specify how to answer questions?

Formal Languages

(why not English, or other natural language)?

- Natural languages are **ambiguous**, for instance:
 - “The fisherman went to the bank”
 - “The boy saw a girl with a telescope”
- Ambiguity makes it difficult to interpret meaning of phrases/sentences
 - But also makes inference harder to define and compute
- Symbolic logic is a syntactically unambiguous language

Propositions

- Propositions are entities (facts or non-facts) that can be true or false

Examples:

- “The sky is blue” - the sky is blue (here and now).
- “Socrates is bald” (assumes ‘Socrates’, ‘bald’ are well defined)
“The car is red” (requires ‘the car’ to be identified)
- “Socrates is bald and the car is red” (complex proposition)
- Use single letters to represent propositions, e.g. P : Socrates is bald
- Reasoning is independent of definitions of propositions

Propositional Logic

- Letters stand for “basic” propositions
- Combine into more complex sentences using operators **not**, **and**, **or**, **implies**, **iff**
- Propositional **connectives**:

\neg negation

$\neg P$

“not P”

\wedge conjunction

$P \wedge Q$

“P and Q”

\vee disjunction

$P \vee Q$

“P or Q”

\rightarrow implication

$P \rightarrow Q$

“If P then Q”

\leftrightarrow bi-implication

$P \leftrightarrow Q$

“P if and only if Q”

From English to Propositional Logic

- “It is not the case that the sky is blue”: $\neg B$
(alternatively “the sky is not blue”)
- “The sky is blue and the grass is green”: $B \wedge G$
- “Either the sky is blue or the grass is green”: $B \vee G$
- “If the sky is blue, then the grass is not green”: $B \rightarrow \neg G$
- “The sky is blue if and only if the grass is green”: $B \leftrightarrow G$
- “If the sky is blue, then if the grass is not green, the plants will not grow”:
 $B \rightarrow (\neg G \rightarrow \neg P)$

First-Order Logic

- Propositional logic has limited expressive power
 - Cannot express relations and ontologies
- **First-Order Logic** can express knowledge about objects, properties and relationships between objects
- Syntax
 - Constant symbols: $a, b, \dots, Mary$ (objects)
 - Variables: x, y, \dots
 - Function symbols: $f, mother_of, sine, \dots$
 - Predicate symbols: $Mother, likes, \dots$
 - Quantifiers: \forall (universal); \exists (existential)

Language of First-Order Logic

- Terms: constants, variables, functions applied to terms (refer to objects)
 - e.g. a , $f(a)$, $mother_of(Mary)$, ...
- Atomic formulae: predicates applied to tuples of terms
 - e.g. $likes(Mary, mother_of(Mary))$, $likes(x, a)$
- Quantified formulae:
 - e.g. $\forall x likes(x, a)$, $\exists x likes(x, mother_of(y))$
 - Second occurrences of x are **bound** by quantifier (\forall in first case, \exists in second) and y in the second formula is **free**

From English to First-Order Logic

- Everyone likes lying on the beach — $\forall x \text{ likes_lying_on_beach}(x)$
- Someone likes Fido — $\exists x \text{ likes}(x, \text{Fido})$
- No one likes Fido — $\neg(\exists x \text{ likes}(x, \text{Fido}))$ (or $\forall x \neg \text{likes}(x, \text{Fido})$)
- Fido doesn't like everyone — $\neg \forall x \text{ likes}(\text{Fido}, x)$
- All cats are mammals — $\forall x (\text{cat}(x) \rightarrow \text{mammal}(x))$
- Some mammals are carnivorous — $\exists x (\text{mammal}(x) \wedge \text{carnivorous}(x))$
- Note: $\forall x A(x) \Leftrightarrow \neg \exists x \neg A(x)$, $\exists x A(x) \Leftrightarrow \neg \forall x \neg A(x)$

Categories, Objects and FOL

- First-order logic can state facts about categories, relating objects to categories or by quantifying over members.

- An object is a member of a category

$$BB_9 \in Basketballs$$

- A category is a subclass of another category

$$Basketballs \subset Balls$$

- All members of a category have common properties

$$(\forall x)(x \in Basketballs \Rightarrow Spherical(x))$$

- Members of a category can be recognised by properties

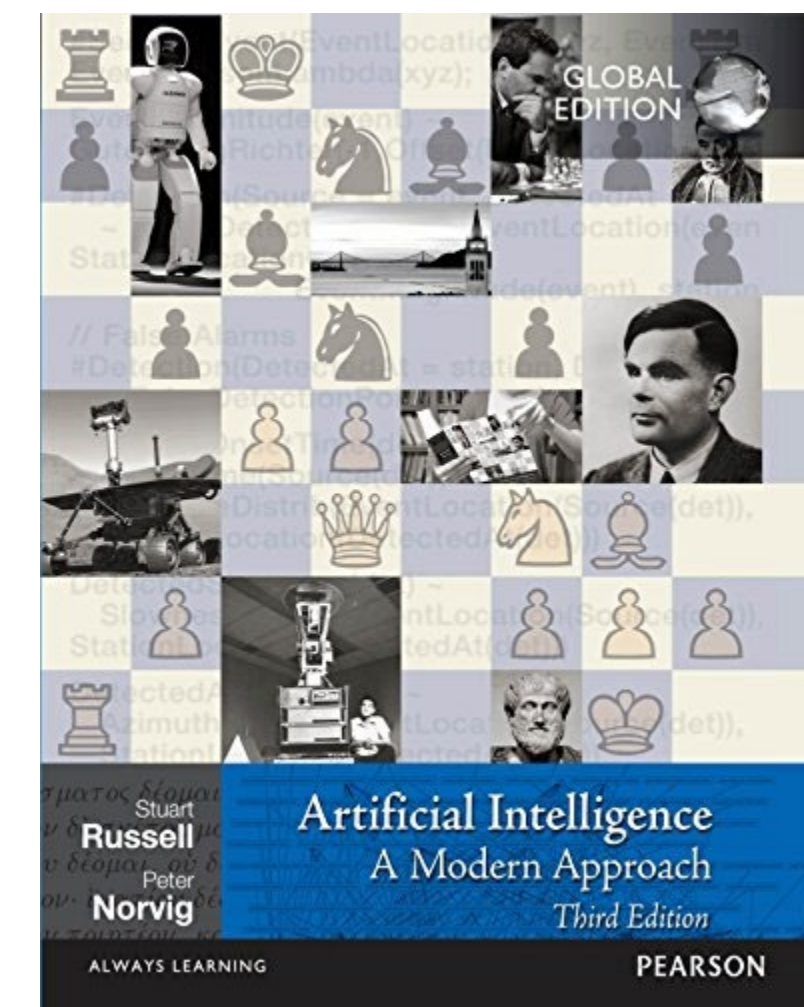
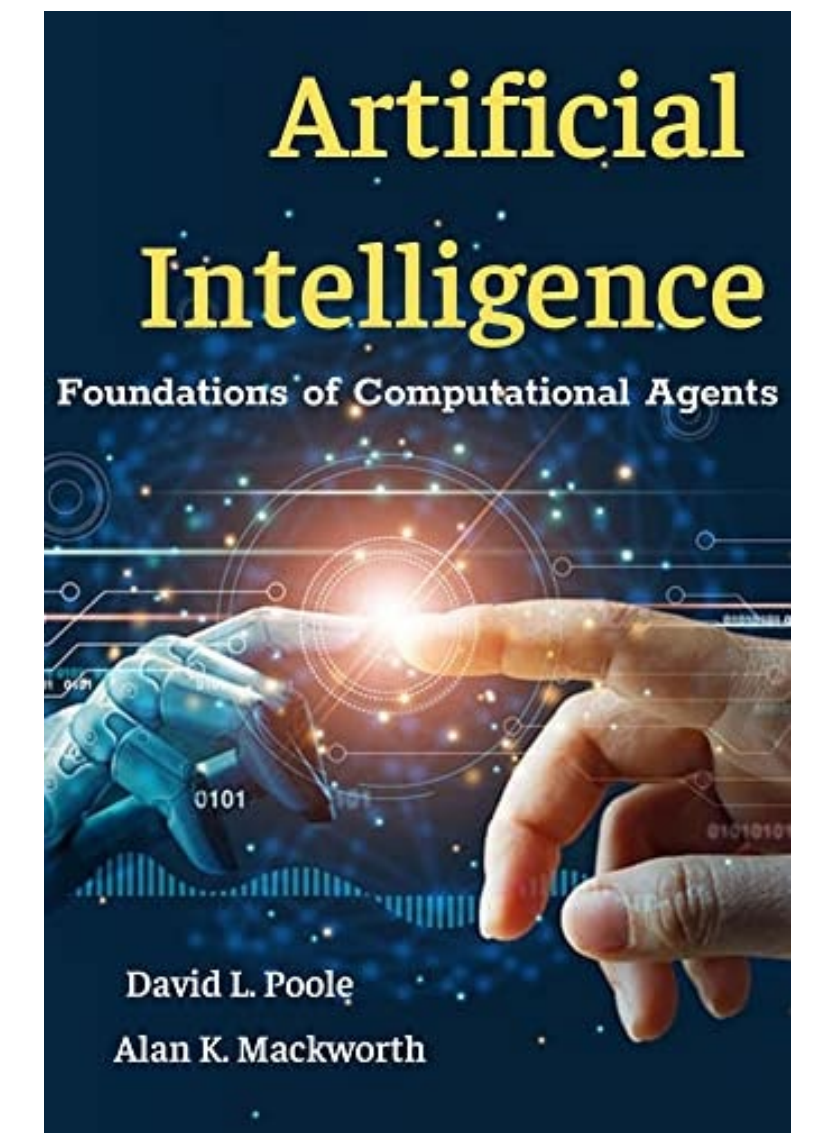
$$Orange(x) \wedge Round(x) \wedge Diameter(x) = 24cm \wedge x \in Balls \Rightarrow x \in Basketballs$$

- A category as a whole has some properties

$$Dogs \in DomesticatedSpecies$$

References


- Poole & Mackworth, Artificial Intelligence: Foundations of Computational Agents, chapter 5.
- Russell & Norvig, Artificial Intelligence: a Modern Approach, chapters 7, 8, 12.



Feedback

- In case you want to provide anonymous feedback on these lectures, please visit:
- <https://forms.gle/KBkN744QuffuAZLF8>

Muchas gracias!



AI Lecture Feedback

This is a short form to provide early feedback for lectures

franciscocruzhh@gmail.com [Switch account](#)

Not shared

* Indicates required question

In case you want a reply, provide your zID. Otherwise your answer is anonymous.

Your answer

how did you participate? *

☐ In the classroom

☐ Watch the class from automatic recording

If you have any comments, feedback, or question about the lectures, this is the place. *

Your answer

Submit Clear form