Symbolic Artificial Intelligence (COMP3008)
Lecture 3: Knowledge-Based Systems
13 October 2025

Daniel Karapetyan
daniel.karapetyan@nottingham.ac.uk

Updates

Moodle quiz 1

- Many of you completed it – well done!
- It is an important part of the learning process; it is particularly helpful if you find it difficult
  - Tracking your learning progress
  - Immediate feedback
  - No pressure; time to revise
  - Prepares you for future lectures and the exam
- There is no expectation that you will correctly answer all the questions in your first attempt but I highly recommend re-attempting it until you get scores close to 100%

Worksheets

- From what I have seen, you work hard and show good progress
  - Worksheets prepare you for the coursework project and the exam
- I would work on them in the natural order; don't skip them
- Worksheet 2 is objectively harder than Worksheet 1
- Read and understand all the text :)
- Using ChatGPT makes no sense; you are not gaining anything
  - Understand someone's answer and compose your own answer are different skills
  - I will not judge you for not understanding something – it's my job to teach you
- I couldn't properly talk to each of you on Friday but we'll have more capacity on Tuesday
- We'll discuss Worksheet 2 at the end of today's lecture
  - Today's lecture is closely related to exercise 7

Knowledge-Based Systems

Expressing knowledge using FOL

Knowledge base does not include an interpretation

- However, it sets restrictions on interpretations

What reasoning can we do?

- Use entailment to prove or disprove a hypothesis (e.g., there is always a green block on top of a non-green block)
- Ask for an interpretation that satisfies our knowledge base (or prove that it does not exist), e.g., to find a list of tracks for a DJ
  - Also the barber's paradox

Example of expressing knowledge

Example of a reasoning query:

- We know that Fido is a dog: we can encode this knowledge as $Dog(fido)$
- We want to know if Fido is a mammal: we can encode this query as $Mammal(fido)$

Can a reasoning system conclude that Fido is a mammal (prove the entailment)?

Example of expressing knowledge

Example of a reasoning query:

- We know that Fido is a dog: we can encode this knowledge as $Dog(fido)$
- We want to know if Fido is a mammal: we can encode this query as $Mammal(fido)$

Can a reasoning system conclude that Fido is a mammal (prove the entailment)?

Predicates *Dog* and *Mammal* are currently unrelated; we can have an interpretation where Fido is not a mammal

As it is, automated reasoning system **cannot** conclude that if $Dog(fido)$ then $Mammal(fido)$

Example of a knowledge base

The key idea of a *knowledge base* (KB) is to explicitly include known connections via sentences, e.g.:

$$\forall x.\mathit{Dog}(x) \rightarrow \mathit{Mammal}(x)$$

Now, if we also know that $\mathit{Dog}(\mathit{fido})$, we can answer our questions, i.e. conclude that $\mathit{Mammal}(\mathit{fido})$:

$$\{\forall x.\mathit{Dog}(x) \rightarrow \mathit{Mammal}(x),\ \mathit{Dog}(\mathit{fido})\} \models \mathit{Mammal}(\mathit{fido})$$

Knowledge base

KB is a set of sentences that are *believed* to be true

- Believed – by the creator of the KB
- The KB-system will assume that the KB is correct

Expressing knowledge in FOL

‘Data types’

- FOL does not support data types; the only data type (apart from Boolean) is the domain of discourse
- To represent real-world objects, we may need to separate objects of different types
- Convenient to use predicates, e.g.

$$Person(x)$$

  is true if and only if x is a person
- OOP analogue: type-of

Examples

- Classification of objects in the domain: dogs, cats, fish, . . .
  How would you define such a knowledge?

Examples

- Classification of objects in the domain: dogs, cats, fish, . . .

  How would you define such a knowledge?

  Predicates $Dog(x)$, $Cat(x)$, $Fish(x)$

Examples

- Classification of objects in the domain: dogs, cats, fish, . . .

  How would you define such a knowledge?

  Predicates $Dog(x)$, $Cat(x)$, $Fish(x)$
- Can have multiple (intersecting) classifications: dark, heavy, aggressive, . . .

  How would this affect the structure of the KB?

Examples

- Classification of objects in the domain: dogs, cats, fish, . . .

  How would you define such a knowledge?

  Predicates $Dog(x)$, $Cat(x)$, $Fish(x)$
- Can have multiple (intersecting) classifications: dark, heavy, aggressive, . . .

  How would this affect the structure of the KB?

  Predicates $Dark(x)$, $Heavy(x)$, $Aggressive(x)$

Examples

- Classification of objects in the domain: dogs, cats, fish, ...

  How would you define such a knowledge?

  Predicates $Dog(x)$, $Cat(x)$, $Fish(x)$
- Can have multiple (intersecting) classifications: dark, heavy, aggressive, ...

  How would this affect the structure of the KB?

  Predicates $Dark(x)$, $Heavy(x)$, $Aggressive(x)$
- Can have hierarchies: mammal, animal, ...

  How would this affect the structure of the KB?

Examples

- Classification of objects in the domain: dogs, cats, fish, ...

  How would you define such a knowledge?

  Predicates $Dog(x)$, $Cat(x)$, $Fish(x)$
- Can have multiple (intersecting) classifications: dark, heavy, aggressive, ...

  How would this affect the structure of the KB?

  Predicates $Dark(x)$, $Heavy(x)$, $Aggressive(x)$
- Can have hierarchies: mammal, animal, ...

  How would this affect the structure of the KB?

  Predicates $Mammal(x)$, $Animal(x)$

Shortcut

In case of a hierarchy, we have to provide several sentences for each object, e.g.:

- $Dog(fido)$
- $Mammal(fido)$
- $Animal(fido)$

This is time-consuming and prone to mistakes

Is there a better solution?

Shortcut

In case of a hierarchy, we have to provide several sentences for each object, e.g.:

- $Dog(fido)$
- $Mammal(fido)$
- $Animal(fido)$

This is time-consuming and prone to mistakes

Is there a better solution?

Add rules such as

- $\forall x.Dog(x) \rightarrow Mammal(x)$
- $\forall x.Mammal(x) \rightarrow Animal(x)$

These rules describe what we believe always to be true

Properties

In OOP, we can have fields to implement composition: has-a

What are our options in FOL?

Function	Predicate
$age(x)$	$Age(x, y)$
e.g. $age(john) = 45$	e.g. $Age(john, 45)$ and $\neg Age(john, y)$ for $y \neq 45$
Imperative languages	Logical languages

Knowledge Base

- Statements such as Age(john, 45) are simple facts
  They are expressed using atomic sentences
- More complex facts may give ‘rules’, e.g.

$$\forall x.\forall y.Father(x,y) \rightarrow \neg Father(y,x)$$

$$\forall x.\forall y.\forall z.Father(x,y) \wedge Father(y,z) \rightarrow GrandFather(x,z)$$

- Question: what if x or y are not people?
- FOL is abstract; we may have to express obvious facts such as the above one

Some common rule types

Subtype: $\forall x.Dog(x) \rightarrow Animal(x)$

Disjoint: $\forall x.\neg(Dog(x) \wedge Cat(x))$

Exhaustive: $\forall x.Dog(x) \rightarrow Alive(x) \vee Dead(x)$

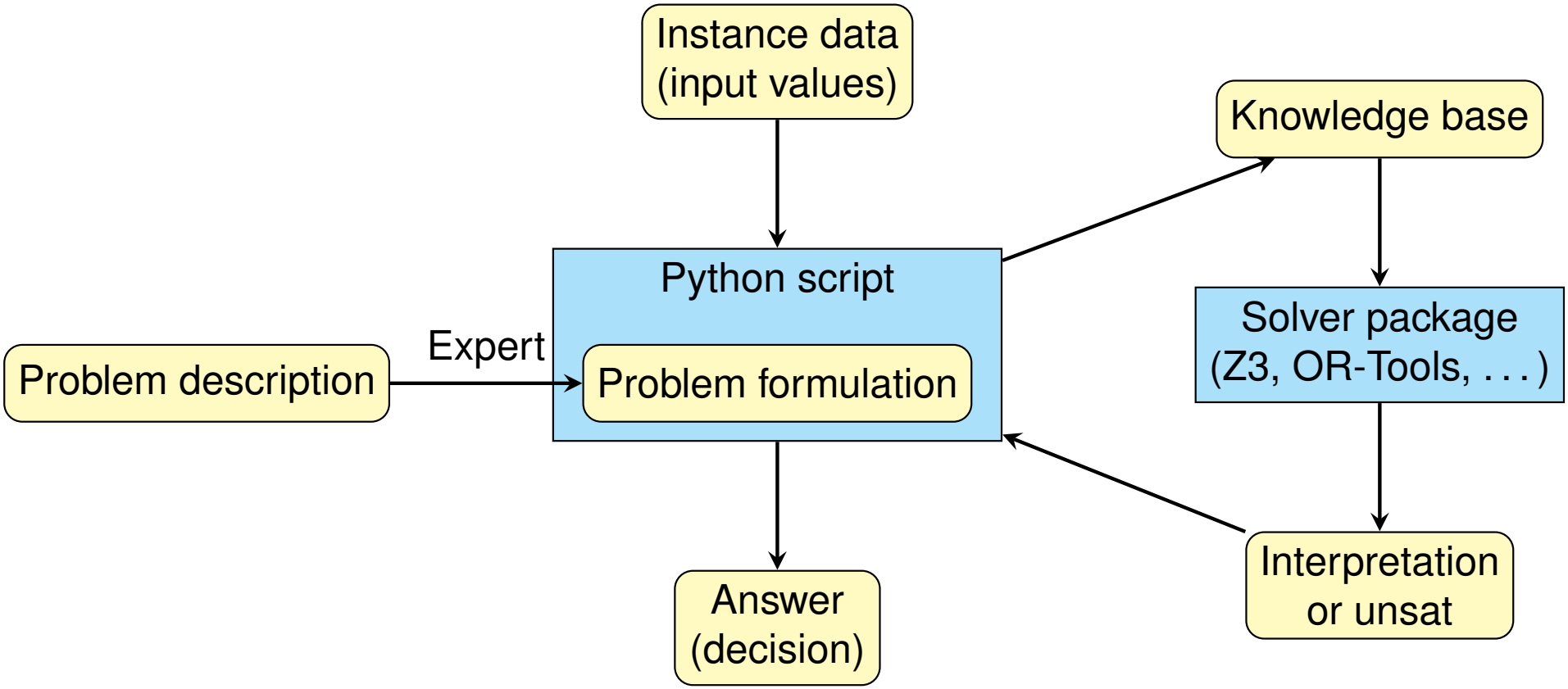
Symmetry: $\forall x.\forall y.Married(x,y) \leftrightarrow Married(y,x)$

Inverse: $\forall x.\forall y.Parent(x,y) \leftrightarrow ChildOf(y,x)$

Type restriction: $\forall x.\forall y.Parent(x,y) \rightarrow Person(x) \wedge Person(y)$

Problem formulations

Real-world knowledge-based systems for decision support



Formulation

- Mathematically describes the problem
- Consists of constraints (knowledge); the solution has to satisfy all the constraints
- Parametrised: one needs instance data to produce a specific knowledge base
- There can be multiple correct formulations of a problem
- The performance of the solver may significantly depend on the chosen formulation

Formulation constraints

Single constraint:

$$\langle \text{FOL formula} \rangle$$

- Example:

$$\exists x.A(x) \rightarrow B(x)$$

Family of constraints – create a constraint for each described combination of parameters:

$$\langle \text{parametrised FOL formula} \rangle \qquad \langle \text{combinations of parameters} \rangle$$

- Example:

$$\exists x.A(x) \rightarrow B(y) \qquad \forall y \in Y$$

- You can use any unambiguous logic when describing the combinations of parameters

Formulation formatting

- Align the formulas
- Align the combinations of parameters
- Number the formulas to be able to reference them (even if you don't reference them yourself)
- A formulation can be split into multiple parts with the text
- Real-world FOL formulations will tend to also define the domain of discourse.

Example:

$$A(x) \qquad \forall x = 1, 3, \ldots, n-1 \tag{1}$$

$$\neg A(x) \qquad \forall x = 2, 4, \ldots, n \tag{2}$$

$$\forall x.B(x) \rightarrow A(x) \tag{3}$$

Examples of KB engineering

Track list problem (from Lab 2)

A DJ composes a playlist of n tracks. The playlist consists of 'slow blocks', i.e. blocks of slow tracks, and 'fast blocks', i.e. blocks of fast tracks. The fast and slow blocks alternate. The DJ wants to follow several rules:

1. The slow block cannot include more than two tracks.
2. The fast block has to include at least three tracks.
3. Tracks 1, 8, 15, 22, ... have to be slow. (Assume indexing from 1.)
4. Tracks 6, 11, 16, 21, ... have to be fast.

Approach

1. What kind of an answer is expected?
2. How can we represent this answer using predicates and functions?
3. Introduce notations
4. Express the knowledge (the rules)

Latin Squares

Latin square is a square matrix M_{ij} of size $n \times n$ filled with elements $\{1, 2, \ldots, n\}$ such that

- each of them occurs exactly once in each row of M_{ij}, and
- each of them occurs exactly once in each column of M_{ij}.

3	1	2
2	3	1
1	2	3

1	3	2
3	2	1
2	1	3

Latin Square Problem

- Given n
- Given some values in M_{ij}
- Task: fill in the rest of the values (or prove unsat).

?	?	1
2	?	?
?	?	3

Latin Squares KB engineering

What kind of an answer is expected?

Latin Squares KB engineering

What kind of an answer is expected?

- An assignment of values to each tile

Latin Squares KB engineering

What kind of an answer is expected?

- An assignment of values to each tile

What are the options to represent the values of all the tiles?

Latin Squares KB engineering

What kind of an answer is expected?

- An assignment of values to each tile

What are the options to represent the values of all the tiles?

1. A constant for each tile
2. Function $tile(x, y)$
3. Predicate $Tile(x, y, v)$
4. ...

Latin Squares KB engineering

Once you decided on the *solution representation*...

- Introduce the notations
- Define the rules: no repeating values in a row, no repeating values in a column
- Define the values of given tiles

The rules are easy to express once the solution representation is defined