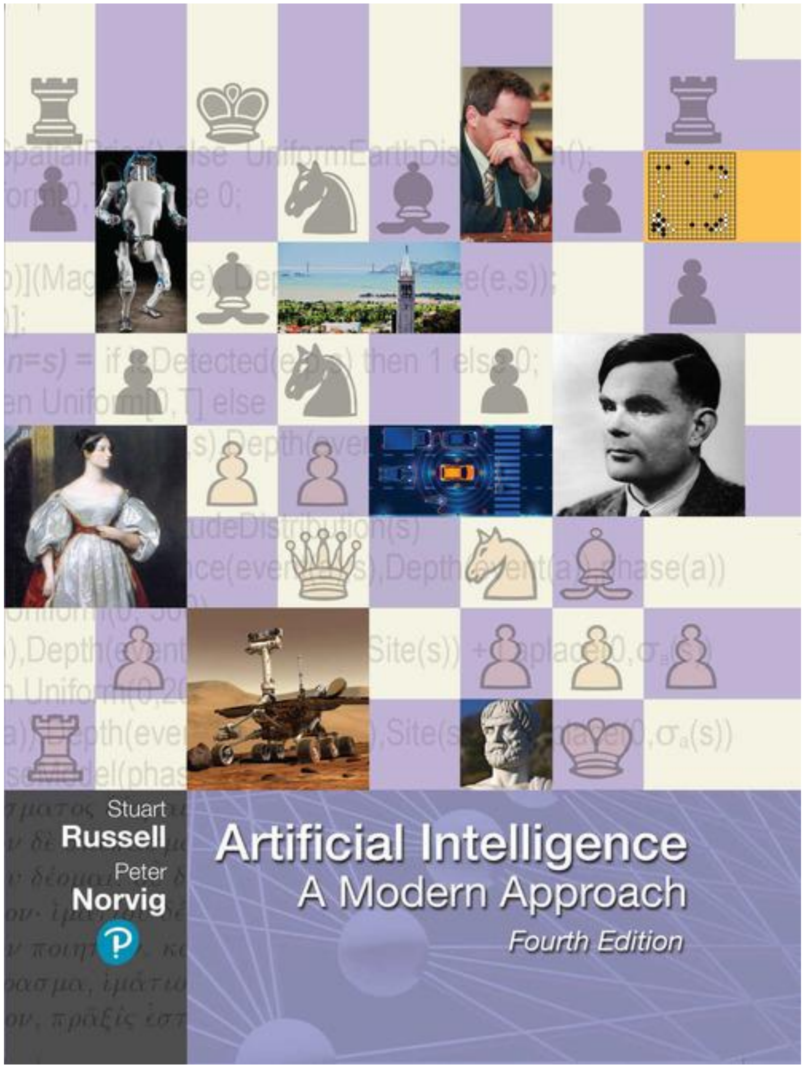
Artificial Intelligence Fundamentals

2024-2025



All opinions are not equal. Some are a very great deal more robust, sophisticated and well supported in logic and argument than others."

– Douglas Adams

AIMA Chapter 8

First-order logic

Outline

- Why FOL?
- Syntax and semantics of FOL
- Fun with sentences
- Wumpus world in FOL
- Knowledge Engineering in FOL

Pros and cons of propositional logic

+ Propositional logic is declarative: pieces of syntax correspond to facts

+ Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)

+ Propositional logic is compositional:
meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

+ Meaning in propositional logic is context-independent (unlike natural language, where meaning depends on context)

- Propositional logic has very limited expressive power (unlike natural language)
E.g., cannot say "pits cause breezes in adjacent squares"
except by **writing one sentence for each square**

First-order logic

Whereas propositional logic assumes world contains facts,
first-order logic (like natural language) assumes the world contains

- **Objects**: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- **Relations**: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- **Functions**: father of, best friend, third inning of, one more than, end of . . .

Pearson

Logics in general

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Syntax of FOL: Basic elements

Constants	$KingJohn,\ \mathbf{2},\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic sentences

Atomic sentence = $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term = $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$

E.g.,

$$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$$

Pearson

Complex sentences

Complex sentences are made from atomic sentences using connectives

$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

Sentences are true with respect to a model and an interpretation

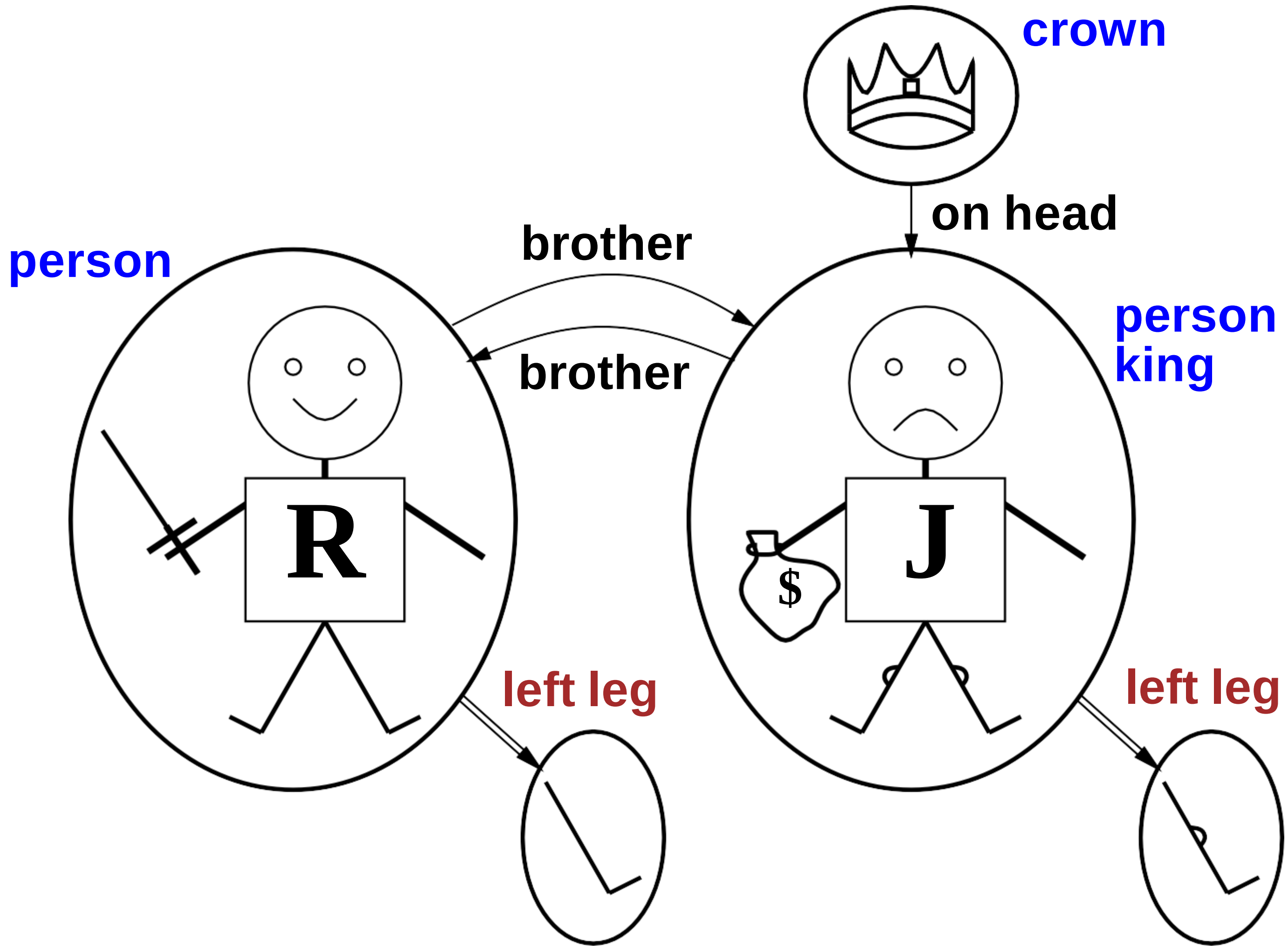
Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols → objects
predicate symbols → relations
function symbols → functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Truth example

Consider the interpretation in which

Richard $\rightarrow$ Richard the Lionheart

John $\rightarrow$ the evil King John

Brother $\rightarrow$ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Pearson

Models for FOL: Lots!

Entailment in propositional logic can be computed by **enumerating models**

We can enumerate the FOL models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞

  For each k-ary predicate P_k in the vocabulary

    For each possible k-ary relation on n objects

      For each constant symbol C in the vocabulary

        For each choice of referent for C from n objects . . .

Computing entailment by enumerating FOL models is not easy!

Universal quantification

$\forall\ \{variables\}\ sentences$

Everyone at Berkeley is smart:
$\forall x\ \ At(x, Berkeley)\ \Rightarrow\ Smart(x)$

$\forall x\ \ P$ is true in a model m iff P is true with x being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \Rightarrow Smart(KingJohn)) \\
\wedge\ & (At(Richard, Berkeley) \Rightarrow Smart(Richard)) \\
\wedge\ & (At(Berkeley, Berkeley) \Rightarrow Smart(Berkeley)) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

Typically, $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x \;\; At(x, Berkeley) \wedge Smart(x)$$

means "Everyone is at Berkeley and everyone is smart"

Existential quantification

$\exists \langle variables \rangle \; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists\, x \;\; At(x, Stanford) \wedge Smart(x)$

$\exists\, x \;\; P$ is true in a model m iff P is true with x being **some** possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Stanford) \wedge Smart(KingJohn)) \\
\vee\;\; & (At(Richard, Stanford) \wedge Smart(Richard)) \\
\vee\;\; & (At(Stanford, Stanford) \wedge Smart(Stanford)) \\
\vee\;\; & \ldots
\end{aligned}
$$

Pearson

Another common mistake to avoid

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists\, x \;\; At(x, Stanford) \;\Rightarrow\; Smart(x)$$

is true if there is anyone who is not at Stanford!

P	Q	$P \to Q$
T	T	T
T	F	F
F	T	T
F	F	T

Properties of quantifiers

$\forall x\ \ \forall y$ is the same as $\forall y\ \ \forall x$

$\exists x\ \ \exists y$ is the same as $\exists y\ \ \exists x$

$\exists x\ \ \forall y$ is not the same as $\forall y\ \exists x$

$\exists x\ \ \forall y\ \ Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y\ \exists x\ \ Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x\ \ Likes(x, IceCream) \qquad \neg\exists x\ \ \neg Likes(x, IceCream)$

$\exists x\ \ Likes(x, Broccoli) \qquad \neg\forall x\ \ \neg Likes(x, Broccoli)$

Fun with sentences

Brothers are siblings

Fun with sentences

Brothers are siblings

$\forall\, x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$.

"Sibling" is symmetric

Fun with sentences

Brothers are siblings

$\forall\, x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$

"Sibling" is symmetric

$\forall\, x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$

One's mother is one's female parent

Fun with sentences

Brothers are siblings

$\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y).$

"Sibling" is symmetric

$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x).$

One's mother is one's female parent

$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y)).$

A first cousin is a child of a parent's sibling

Fun with sentences

Brothers are siblings

$\forall x, y \ \ Brother(x, y) \ \Rightarrow \ Sibling(x, y)$.

"Sibling" is symmetric

$\forall x, y \ \ Sibling(x, y) \ \Leftrightarrow \ Sibling(y, x)$.

One's mother is one's female parent

$\forall x, y \ \ Mother(x, y) \ \Leftrightarrow \ (Female(x) \wedge Parent(x, y))$.

A first cousin is a child of a parent's sibling

$\forall x, y \ \ FirstCousin(x, y) \ \Leftrightarrow \ \exists\, p, ps \ \ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall x \;\times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
$2 = 2$ is valid

E.g., definition of (full) *Sibling* in terms of *Parent*:

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \wedge \exists m, f \;\; \neg(m = f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$$

Interacting with FOL KBs

Suppose a wumpus-world agent is using a FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$:

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists\, a \;\; Action(a, 5))$

I.e., does KB entail any particular actions at $t = 5$?

Answer: $Yes, \; \{a/Shoot\}$ ← substitution (binding list)

Given a sentence S and a substitution σ,

$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Pearson

Knowledge base for the wumpus world

"Perception"

$\forall\, b, g, t \;\; Percept([Smell, b, g], t) \;\Rightarrow\; Smelt(t)$

$\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$

Reflex: $\forall\, t \;\; AtGold(t) \;\Rightarrow\; Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \;\; AtGold(t) \land \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Properties of locations:

$\forall x, t \;\; At(Agent, x, t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$

$\forall x, t \;\; At(Agent, x, t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \;\; Pit(x) \land Adjacent(x, y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition for the *Breezy* predicate:

$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x, y)]$$

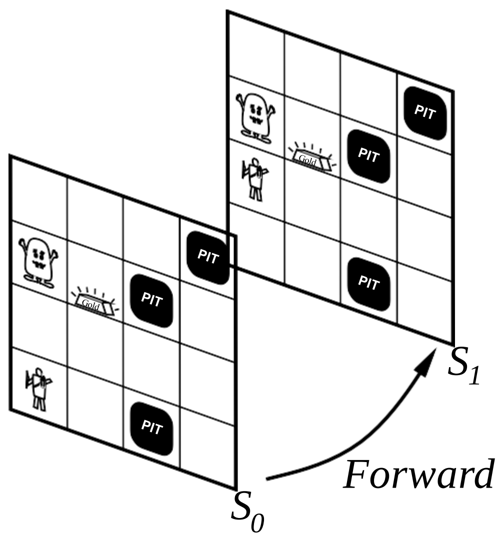
Keeping track of change

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:

  Adds a situation argument to each non-eternal predicate
  E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a in s



Describing actions I

"Effect" axiom—describe changes due to action

$\forall s \; AtGold(s) \Rightarrow Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action

$\forall s \; HaveArrow(s) \Rightarrow HaveArrow(Result(Grab, s))$

Frame problem: find an elegant way to handle non-change

(a) representation—avoid frame axioms

(b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is "about" a predicate (not an action per se):

P true afterwards $\Leftrightarrow$ [an action made P true
$\lor$ P true already and no action made P false]

For holding the gold:

$$\forall a, s \;\; Holding(Gold, Result(a, s)) \Leftrightarrow$$
$$[(a = Grab \land AtGold(s))$$
$$\lor\ (Holding(Gold, s) \land \neg(a = Release))]$$

Making plans

Initial condition in KB:

$At(Agent, [1, 1], S_0)$
$At(Gold, [1, 2], S_0)$

Query: $Ask(KB, \exists\, s\ \ Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Knowledge Engineering in FOL

Knowledge engineering: *the general process of knowledge-base construction.*

The steps used in the knowledge engineering process:

1. Identify the questions.
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the problem instance
6. Pose queries to the inference procedure and get answers
7. Debug and evaluate the knowledge base

A complete example is available in the reference book at the end of chapter 8.

Summary

First-order logic:

– objects and relations are semantic primitives
– syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:

– conventions for describing actions and change in FOL
– can formulate planning as inference on a situation calculus KB

Developing a KB in FOL requires a careful process of analyzing the domain, choosing a vocabulary, and encoding the axioms required to support the desired inferences.

AIMA notebooks: «logic.ipynb»

Logic

This Jupyter notebook acts as supporting material for topics covered in **Chapter 6 Logical Agents**, **Chapter 7 First-Order Logic** and **Chapter 8 Inference in First-Order Logic** of the book *Artificial Intelligence: A Modern Approach*. We make use of the implementations in the logic.py module. See the intro notebook for instructions.

Let's first import everything from the `logic` module.

In [1]:

```
from utils import *
from logic import *
from notebook import psource
```

CONTENTS

- Logical sentences
  - Expr
  - PropKB
  - Knowledge-based agents
  - Inference in propositional knowledge base
    - Truth table enumeration
    - Proof by resolution
    - Forward and backward chaining
    - DPLL
    - WalkSAT
    - SATPlan
  - FolKB
  - Inference in first order knowledge base
    - Unification
    - Forward chaining algorithm
    - Backward chaining algorithm

https://github.com/aimacode/aima-python/blob/master/logic.ipynb

In the next lecture...

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward and backward chaining
- Logic programming
- Resolution