Artificial Intelligence:

First-Order Logic

Course V231
Department of Computing
Imperial College, London

Jeremy Gow

AI & Logic

- Logic is a good **KR language** (lecture 4)
  - ★ provides formal basis for many AI techniques
- Not the only approach!
  - ★ alternative **symbolic** KR schemes
  - ★ **sub-symbolic** approaches
    - e.g. neural networks (see lectures 12 & 13)

Logic-Based Agents

- Knowledge base (KB)
  - Set of logical sentences describing environment
  - Adds new sentences via...
    - Observation of environment
    - Reasoning with existing knowledge
- Logical reasoning
  - **Deduction**: can X be explained by what I know?
  - **Abduction**: what fact would explain X?
  - **Induction**: what rule would explain X?

Logical Reasoning

- **Deduction** (mathematician)
  - {A, A→B} so B
- **Abduction** (detective)
  - {B, A→B} so A
- **Induction** (scientist)
  - Always see B with A so A→B

Only deduction is guaranteed

Automated Reasoning

- Enables logic-based AI agents
- Usually refers to **automated deduction**
  - This and next three lectures
  - Basis for **logic programming**, e.g. Prolog

- Also logic-based **machine learning** (induction)
  - See later lectures (inc. inductive LP)
- And **automated abduction**
  - Outside this course (as is abductive LP)

First-Order Logic

- Central to automated deduction in AI
- This lecture
    - Syntax & Semantics
        - Propositional then first-order
    - From FOL to Prolog

- Lecture 7: Deduction in FOL
- Lecture 8-9: Automated deduction in FOL

Propositional Syntax

- **Constants**: true, false, (T/⊥, T/F, 1/0) “truth values”
- **Variables**: represent propositions (P, Q, ...)
- **Brackets**: (and)
- **Connectives**

  $\neg$ not (negation) also ~

  $\wedge$ and (conjunction) also & , . (or just PQ)

  $\vee$ or (disjunction) also | ; +

  $\rightarrow$ if then (implication) also $\leftarrow$ $\Rightarrow$ $\supset$

  $\leftrightarrow$ if & only if (equivalence) also $\Leftrightarrow$ $\equiv$

Propositional Sentence

A **sentence** is either

1. A constant or variable
2. $\neg P$, $P \wedge Q$, $P \vee Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ or (P) for formulae P and Q

$(A \wedge B) \leftrightarrow (B \wedge A)$ ✔

$A \wedge \neg A$ ✔

$\neg \wedge A \rightarrow A \rightarrow$ ✘

Propositional Semantics

- Which **possible worlds** is a sentence true?
- A **model** defines a possible world
  - ★ Assigns true or false to each variable
- **Truth table** below define connectives
  - ★ Now know *truth of a sentence for any model*

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Using Truth Tables

- When is $(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$ true?
- Draw truth table showing subformulae vs models
  - ★ Brackets: good places to stop and work out

P	Q	$P \rightarrow Q$	$\neg P \vee Q$	$\ldots \leftrightarrow \ldots$
false	false	true	true	**true**
false	true	true	true	**true**
true	false	false	false	**true**
true	true	true	true	**true**

- It is always true: it is a **valid** sentence (see L7)

Propositional Pitfalls

- False implies anything
  - P: “5 is even”, Q: “7 is even”
  - $P \rightarrow Q$ is false $\rightarrow$ false is true
- Implication does not indicate relevance
  - P: “5 is odd”, Q: “Tokyo is capital of Japan”
  - $P \rightarrow Q$ is true $\rightarrow$ true is true

More Propositional Pitfalls

- Lecture 4 example: “Every Monday **and** Wednesday I go to John’s house for dinner”

- M = Is Monday, W = Is Wednesday
- J = I go to John’s house for dinner

$$M \vee W \rightarrow J$$

- ‘and’ became ‘or’
- $\vee$ is not exclusive: can be Monday **and** Wednesday

First-Order Syntax

- A **term** is
  - a constant (lower-case), e.g. apple, red
  - a variable (upper-case), e.g. X, Y, ...
  - a function applied to terms, e.g. colour(apple)
- A **proposition** is: a predicate applied to terms
- A **formula** is propositions combined with
  - propositional connectives (as before)
  - quantifiers $\forall$, $\exists$
- A **sentence** is a ‘properly quantified’ formula

Ground Terms

- Constants **directly** represent objects
  - physical objects (apple) or concepts (red)
- Functions **indirectly** represent objects
  - father(john), colour(apple), (1 + 2) + 3
- **Arity** = number of arguments of function
  - unary, binary, ... n-ary

- **Ground terms** = constants + functions + brackets
  - They represent specific objects

Semantics of Ground Terms

- A first-order **model** is a pair (Δ, Θ)
  - Δ is a **domain**, a non-empty set
  - Θ is an **interpretation**, associates ground terms with elements of Δ
- $\Theta(c) \in \Delta$ for constant, $\Theta(f): \Delta^n \rightarrow \Delta$ for n-ary f
- So every ground term mapped to an element of Δ
  - father(john) and jack are different terms
  - but Θ could map them to same element

Unwanted Terms

- No restrictions on applying functions
    - Things we want like father(john) and 1 + 1
    - But also father(1) and red + john
- **Solution 1**: arbitrary interpretation (father(1) to 0)
- **Solution 2**: Δ includes undefined element
    - Θ maps unwanted applications to undefined
    - Function with undef. argument is undef.
- **Practical approach**: avoid using these terms
- Other solutions in type theory, multi-valued logics,...

Predicates

- **Predicates** are relationships between objects
  - Relate ground terms: brother(bob, father(bill))
  - Define arity as before (brother is binary)
  - A proposition that is true or false
- Semantics defined by interpretation Θ
  - $\Theta(p): \Delta^n \rightarrow \{true, false\}$ for predicate p
  - False when any argument is undefined
- = (equality) is a predicate

Ground Formulae

- Ground formulae = ground terms + predicates + connectives
  - Statements about specific objects
  - Model (Δ, Θ) tells us whether true or false

  $(1 < 2) \wedge (3 < 2)$

  $lectures(simon, ai) \wedge lectures(jeremy, ai)$

  $odd(3 + 2) \rightarrow capital(japan, tokyo)$

  $brother(bob, father(bill)) \vee \neg mother(jane, bill)$

- pred and ¬pred are **positive** and **negative literals**

Choosing Predicates & Terms

- "The cost of an omelette at the Red Lion is £5"

  cost_at_red_lion_is(omelette, five_pounds)

  cost_is(omelette, red_lion, five_pounds)

  cost(omelette, red_lion) = five_pounds

- "Omelettes cost less than pies"

  $cost\_is(omelette, a) \wedge cost\_is(pie, b) \wedge a < b$

  $cost(omelette) < cost(pie)$

Variables

- Variables allow us to talk about objects in general
- A variable is just another kind of term
  - Terms: father(X), (X + 2) + Y
  - Propositions: brother(X, father(Y)), X < (Y + 1)
  - Formulae: lectures(X, ai) ∧ lectures(Y, ai), cost(X, red_lion) = 3

★ Variables make formulae etc. **non-ground**

Quantifiers

- For formula f with $X \in Var(f)$ (X is free variable)
  - $\forall X.f$ is a formula (universal quantification)
  - $\exists X.f$ is a formula (existential quantification)
- **Free variables** Var defined by
  - Var(f) = all variables, for quantifier-free f
  - $Var(\forall X.\ f) = Var(\exists X.f) = Var(f) - \{X\}$

$$Var(X < Y + 1) = \{X, Y\}$$

$$Var(\exists X.\ X < Y + 1) = \{Y\} \quad (\text{"X is bound"})$$

Semantics of Quantifiers

- **Substitution** $\{t/X\}$ replaces X with term t
  - f.$\{t/X\}$ is a formula f with free var X replaced
- For model (Δ, Θ)
  - $\forall$X. f is true **iff** f.$\{t/X\}$ is true for all t in Δ
  - $\exists$X.f is true **iff** f.$\{t/X\}$ is true for some t in Δ
- Formulae = terms + predicates + connectives + quantifiers
  - A **sentence** is a formula with no free variables
  - Only sentences are true or false for a model

Translation Pitfalls

- "There is a meal at the Red lion which cost £3"

$$\exists X.(\mathrm{meal}(X) \wedge \mathrm{cost}(X, \mathrm{red\_lion}) = 3$$

- "All the meals at the Red lion cost £3" ($\exists$ to $\forall$?)

$$\forall X.(\mathrm{meal}(X) \wedge \mathrm{cost}(X, \mathrm{red\_lion}) = 3) \quad ✘$$

$$\forall X.(\mathrm{meal}(X) \rightarrow \mathrm{cost}(X, \mathrm{red\_lion}) = 3) \quad ✘$$

$$\forall X.(\mathrm{meal}(X) \wedge \mathrm{serves}(\mathrm{red\_lion}, X) \rightarrow \mathrm{cost}(X, \mathrm{red\_lion}) = 3 \quad ✔$$

- Be careful with order and type of quantifiers

Prolog

- Declarative programming language
  - Not procedural
  - Tell **what** to compute not **how** to compute

- Logic programming
  - Algorithm = Logic + Control (Kowalski)
    - ★ Logic: the representation
    - ★ Control: the search techniques
- Prolog = (FOL Horn clauses) + (SLD resolution)

Horn Clauses

- **Horn clauses** are subset of FOL sentences

  $$\forall X_1 \ldots \forall X_n.((P_1 \wedge \ldots \wedge P_m) \rightarrow H)$$

  for positive literals P_i and H
- $P_1 \wedge \ldots \wedge P_m$ is the body, H is the head
- For m = 0 this is a **fact** $\forall X_1 \ldots \forall X_n.H$
- Often assume $\forall$s
  - brother(X, Y) $\wedge$ father(Y, Z) $\rightarrow$ uncle(X, Z)
  - brother(john, X)

Prolog Programs

- A Prolog program is a list of FOL Horn clauses
- Translate each clause...
  1. Drop universal quantifiers: $P_1 \land \ldots \land P_m \rightarrow H$
  2. Rotate around implication: $H \leftarrow P_1 \land \ldots \land P_m$
  3. Write $\leftarrow$ as :- and $\land$s as commas, and always end with a full stop: `H :- P1, ... , Pm.`

Example Translation

- "If the lecture has a good lecturer and the subject is interesting then students are awake and listening"

  $\forall X.(good\_lecturer(X) \wedge interesting\_subject(X) \rightarrow students\_awake(X))$

  $\forall X.(good\_lecturer(X) \wedge interesting\_subject(X) \rightarrow students\_listening(X))$

- In Prolog:

```
students_awake(X) :- good_lecturer(X), interesting_subject(X)
students_listening(X) :- good_lecturer(X), interesting_subject(X)
```

Search in Prolog

- Given a Prolog program (database of Horn clauses) and a query $q(t_1, ..., t_n)$
- Scans database for clauses with n-ary q as head
  - For head $q(s_1, ..., s_n)$ tries to find variable assignment such that $t_i = s_i$ (see next lecture)
  - Using variable assignment any literals in body of clause becomes new queries
- Succeeds when all queries **proved** (via facts)
  - Returns variable assignment for original query

Negation As Failure

- Given knowledge pm(thatcher) and pm(blair) we can't prove pm(brown), but it might be true
- Alternately, make the **closed world assumption**
  - If we can't prove it then it is false
  - So pm(brown) is false

- Prolog uses **negation as failure** based on CWA
  - A query \+q is proved by failing to prove q

Prolog Search Example

```
uncle(U, N) :-  brother(U, F), father(F, N).
uncle(U, N) :-  brother(U, M), mother(M, N).
brother(bob, bill).
brother(barry, betty).
father(bill, bruce).
mother(betty, bruce).

?- uncle(X, bruce).
X = bob

?- \+ uncle(brian, bruce).
Yes
```

Parallelism and Prolog

- Two processors P1 and P2
- OR-parallelism for query `uncle(X, bruce).`
  - P1 takes `uncle(U, N) :- brother(U, F), father(F, N).`
  - P2 takes `uncle(U, N) :- brother(U, M), mother(M, N).`
- AND-parallelism
  - Use `uncle(X, bruce) :- brother(X, F), father(F, bruce).`
  - P1 takes `brother(X, F)`
  - P2 takes `father(F, bruce)`
  - Trickier because values for `F` must agree

More on Prolog

- See online notes (and Russell & Norvig):
  - How arithmetic is carried out
  - How performance is measured
    - Logical Inferences per Second (LIPS)
  - How performance is improved
    - By compiling code, e.g. to WAM
    - LIPS now in millions
  - Case study: an **expert system** in Prolog