COMP44[illegible]nowledge Representation and Reas[illegible]g

Propositional Logic 1

Maurice Pagnucco
School of Computer Science and Engineering

COMP4418, Week 1

Knowledge Representation and Reasoning

- A knowledge-based agent has at its core a *knowledge base*
- A knowledge base is a set of facts about the domain in which the agent finds itself, expressed in a suitable representation
- These facts are called *sentences*
- Sentences are expressed in a (formal) *knowledge representation language*
- **Question:** How do we write down knowledge about a domain/problem? Once we have written down knowledge, can we automate or mechanise reasoning to deduce new facts?
- References:
  - Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall International, 1995. (Chapter 6)

Overview

- The need for formal Knowledge Representation
- Propositions
- Formulae in Propositional Logic
- Syntax
- Semantics
- The Notion of Proof
- Conclusion

Knowledge Representation and Reasoning

- A knowledge-based agent can be described on different levels:
  - Knowledge level (epistemological level)
  - Logical level
  - Implementation level
- *Knowledge Representation* is concerned with expressing knowledge in a computer-tractable way
- *Reasoning* attempts to take this knowledge and draw inferences (e.g., answer queries, determine facts that follow from the knowledge base, etc.)

Why do we need formal Knowledge Representation?

- You will all be familiar with sentences in English like:
  "The boy saw a girl with a telescope"

  Also:
  "Our shoes are guaranteed to give you a fit" (lexical ambiguity)
  "I heard about him a lot" (structural ambiguity)
  "As he uttered the all-important word he dropped his voice, but she just managed to catch it" (ambiguity of cross reference)

- Natural languages exhibit ambiguity
- Not only does ambiguity make it difficult for us to understand what is the intended meaning of certain phrases and sentences but they also make it very difficult to make any inferences
- We shall investigate symbolic logic as a form of knowledge representation and reasoning

Syntax vs Semantics

Syntax Describes the legal sentences in a knowledge representation language (e.g., in the language of arithmetic expressions $x < 4$).

Semantics Refers to the meaning of sentences. It determines the facts in the world referred to by the sentences in a knowledge representation language. Essentially it relates syntactic descriptions to the "real world". Semantics talks about *truth* and *falsity*. (E.g., $x < 4$ is true when x is a strictly smaller number than 4 and false otherwise)

Note: a sentence does not mean anything by itself. It is up to the person who wrote the sentence to give it a meaning. To do this the person has to provide an *interpretation* for the sentence. In English, interpretations are already fixed but they still needed to be supplied at some point in the past.

Propositions

- Propositions are statements of fact.
- The English sentence “The sky is blue” expresses the proposition that the sky is blue
- Other propositions:
  - *“Socrates is bald”*
  - *“The car is red”*
  - *“The text is readable”, etc.*
- Often we shall use single letters to represent propositions:
  P: Socrates is bald
  (it saves a lot of time!)
- This is referred to as a *scheme of abbreviation*

Scheme of Abbreviation

Q: the lectures are dull
T: the text is readable
P: Alfred will pass

S: the lectures are dull
Q: the lectures are dull
T: the text is readable
P: Alfred will pass

Q: the lectures are dull
Q: the text is readable
P: Alfred will pass

Formulae in Propositional Logic

- Now that we have propos[illegible] would be nice to combine them into more complex expressions (or [illegible]s)
- Similarly to natural languages, we can do so through the use of *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 Formulae

- "it is not the case that the lectures are dull": $\neg Q$ (alternatively "the lectures are not dull")
- "the lectures are dull and the text is readable": $Q \wedge T$
- "either the lectures are dull or the text is readable": $Q \vee T$
- "if the lectures are dull, then the text is not readable": $Q \rightarrow \neg T$
- "the lectures are dull if and only if the text is readable": $Q \leftrightarrow T$
- "if the lectures are dull, then if it is not the case that the text is readable, then it is not the case that Alfred will pass": $Q \rightarrow (\neg T \rightarrow \neg P)$

Formulae in Propositional Logic — Syntax

More formally, we can specify the BNF grammar for formulae in propositional logic as follows:

Sentence ::= *AtomicSentence* | *ComplexSentence*
AtomicSentence ::= **True** | **False** | *P* | *Q* | *R* | ...
ComplexSentence ::= (*Sentence*)
| *Sentence Connective Sentence*
| ¬ *Sentence*
Connective ::= ∧ | ∨ | → | ↔

Semantics

- The semantics of the connectives can be given by *truth tables*

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
True	True	False	True	True	True	True
True	False	False	False	True	False	False
False	True	True	False	True	True	False
False	False	True	False	False	True	True

- Now we have a way of determining the semantics (i.e., truth value) for complex formulae. Just use the truth tables!

Semantics — Complex Formulae

R	S	$\neg R$	$R \wedge S$	$\neg R \vee S$	$(R \wedge S) \rightarrow (\neg R \vee S)$
True	True	False	True	True	True
True	False	False	False	False	True
False	True	True	False	True	True
False	False	True	False	True	True

The Notion of Proof

Question: Now that we can write knowledge down, how do we automate reasoning (i.e., perform inference)?

- A *proof* of a formula from a set of premises is a sequence of steps in which any step of the proof is:
  1. An axiom or premise
  2. A formula deduced from previous steps of the proof via some *rule of inference*

  The last line of the proof should be the formula we wish to prove
- This is intended to formally capture the notion of proof that is commonly applied in other fields (e.g., mathematics)
- We use the notation $\lambda \vdash \rho$ to denote that the formula(s) λ "prove" the formula(s) ρ. Alternatively, we say that ρ *follows* from (premises) λ

What is a Logic?

- A logic consists of
  1. A *formal system* for expressing knowledge about a domain consisting of
     - Syntax Sentences (well-formed formulae)
     - Semantics Meaning
  2. A *proof theory* — rules of inference for deducing sentences from a knowledge base

Terminology

- A sentence is *valid* or necessarily true if and only if it is true under all possible interpretations in any possible world (e.g., $P \vee \neg P$).
  Put another way, a sentence is valid if and only if no matter what truth values are assigned to its constituent parts, the sentence is always true
- Valid sentences are also referred to as *tautologies*
- A sentence is *satisfiable* if and only if in some possible world there is some interpretation for which the sentence is true
- A sentence is *unsatisfiable* if and only if it is not satisfiable (e.g., $P \wedge \neg P$)

Provability

- $\lambda \vdash \rho$ — we can construct a proof for ρ from λ using axioms and rules of inference
- For example, $P,\ P \rightarrow Q \vdash Q$
- This is a syntactic notion involving pure symbol manipulation
- In fact, we can capture $\vdash$ via an inference procedure (several exist)
- If λ is empty (i.e., $\emptyset \vdash \rho$) and ρ is a single formula, then we say that ρ is a *theorem* of the logic

Entailment

- $\lambda \models \rho$ — whenever the formula(s) in λ are true, one of the formula(s) in ρ is true
- This is a semantic notion; it concerns the notion of truth
- In the case where ρ is a single formula, we can determine whether $\lambda \models \rho$ by constructing a truth table for λ and ρ. If, in any row of the truth table where all the formulae in λ are true, ρ is also true, then $\lambda \models \rho$.
- If λ is empty, we say that ρ is a *tautology*

Entailment

P	$P \to Q$	Q
True	True	True
True	False	False
False	True	True
False	True	False

Therefore, $P,\ P \to Q \models Q$

Entailment — Tautologies

R	S	$\neg R$	$R \wedge S$	$\neg R \vee S$	$(R \wedge S) \rightarrow (\neg R \vee S)$
True	True	False	True	True	True
True	False	False	False	False	True
False	True	True	False	True	True
False	False	True	False	True	True

Therefore, $\models (R \wedge S) \rightarrow (\neg R \vee S)$

Soundness and Completeness

- An inference procedure ($\vdash$ for a logic) is *sound* if and only if it preserves truth
- In other words $\vdash$ is sound iff whenever $\lambda \vdash \rho$, then $\lambda \models \rho$
- A logic is *complete* if and only if it is capable of proving all truths
- In other words, whenever $\lambda \models \rho$, then $\lambda \vdash \rho$
- A logic is *decidable* if and only if we can write a mechanical procedure (computer program) which when asked $\lambda \vdash \rho$ it can eventually halt and answer "yes" or answer "no"

Conclusion

- Due to the ambiguity in natural languages there is a need to specify knowledge through the use of formal languages
- Not only will these formal languages give us a way to remove ambiguity but they will also help to provide methods for automating inference
- Propositional logic is a first move in this direction
- In the next lecture we look at automating inference using the *resolution* rule on which Prolog is based
- We shall also investigate first-order predicate calculus
- Keep in mind that there are a large number of logics and knowledge representation schemes that we shall not look at