Topic 2: Propositional logic

How to we **explicitly** represent our knowledge about the world?

References:

Dean, Allen, Aloimonos, Chapter 3
Russell and Norvig: Chapter 6

One of two or three logical languages we will consider.

Logical languages are analogous to programming languages: systems for describing knowledge that have a rigid syntax.

Logical languages (unlike programming languages) emphasize syntax. In principle, the semantics is irrelevant (in a narrow sense).

Knowledge Representation

- Most programs are a set of procedures that accomplish something using rules and “knowledge” embedded in the program itself.
- This is an example of

  implicitly encoded information

  – If you want to change the way Microsoft Word implements variables in macros, you have to hack the code.
  – When my tax program needs to be upgraded for a new tax rule, the code needs to be rewritten.
  – In contrast, when my accountant incorporates the same new rule, little of no brain surgery is required.

Explicit knowledge

When we encode rules in a separate rule book or

Knowledge Base (KB)

we have

explicitly encoded

(some of) the information of interest.

i.e. the rules are separate from the procedures for interpreting them.

– Explicit knowledge encoding, in general, makes it easier to update and manipulate (assuming the encoding is good).

Q. Is a “plug-in” implicit or explicit knowledge?

Knowledge and reasoning

Objective: to <u>explicitly</u> represent knowledge about the world.

- So that a computer can use it efficiently….
  - Simply to use the facts we have encoded
  - To make **inferences** about things it doesn't know yet
- So that we can easily enter facts and modify our knowledge base.

- The combination of a formal language and a reasoning mechanism is a **<u>logic</u>**.
- Each fact: encoded as a <u>sentence</u>.

Wff's

- In practice, with logical languages we combine symbols to express truths, or relationships, about the world.
- If we put the symbols together in a permitted way, we get a **well-formed formula** or **wff**
- A <u>proposition</u> is another term for an allowed formula.
- A **<u>propositional variable</u>** is a proposition that is atomic: that it, it cannot be subdivided into other (smaller) propositions.
- We can combine propositional variables into compound statements (wffs) using <u>truth-functional connectives</u>.

AND, OR, NOT, IMPLIES, EQUIVALENCE

- Formulae are made from propositional variables and the connectives.

Terminology

- A set of wffs connected by AND's is a **conjunction**.
- A set of wffs connected by OR's is a **disjunction**.
- **Literals** plain propositional variables, or their negations: P and ¬ P.

Semantics

- We attach meaning to wffs in 2 steps: 1. By assigning truth values to the propositional variables 2. By associating real-world concepts with symbols
- Step 1, assigning truth values, is called an **interpretation**.
- Step 2 is called **symbol grounding**, and is not related to the logical consistency or mathematical soundness of the logical system.

Discovering "new" truths

- Want to be able to generate new sentences that must be true, given the facts in the KB.
- Generation of new true sentences from the KB is called

  entailment.

- We do this with an **inference procedure.**
- If the **inference procedure** works "right": only get entailed sentences. Then the procedure is **sound** or **truth-preserving**.
  - Q. Why would we ever consider any other kind of inference?

Knowing about knowing

- We would like to have knowledge both about the world, as well as the s*tate of our own knowledge* (i.e. **meta-knowledge**).

- **Ontological commitments** refer to the guarantees given by our logic and KB regarding the real world.
- **Epistemological commitments** relate to the states of knowledge, or kinds of knowledge, that a system can represent.

A particular set of truth assignments associated with propositional variables is a **model** IF THE ASSOCIATED FORMULA (or formulae) come out with the value true.

e.g. For the formula

```
        (A and B) implies ( C and D)
```

the assignment

```
            A=true B=true C=true D=true
```

is a model.

The assignment

```
           A=false B=true C=true D=true
```

is another model, but the assignment

```
           A=true B=true C=true D=false
```

is not a model.

Satisfiability

- If *no model is possible * for a formula, then the formula is NOT **SATISFIABLE**, otherwise it is satisfiable.
- A **<u>Theory</u>** is a set of formulae (in the context of propositional logic).
- If no model is possible for the negation of a formula, then we say the original formula is **<u>valid</u>** (also a formula is always true, it is a ***tautology)***.
- An axiom is a wff that states a priori information.
- Proper axioms state facts.

Completeness

- The set of steps used by a sound procedure to generate new sentences is a **proof**.
- If it is possible for find a proof for any sentence that is entailed, then the inference procedure is **complete**.

- A set of rules is **refutation complete**: *if a set of sentences cannot be satisfied, then resolution will derive a contradiction.* I.e.we can derive both *P* and *not(P)* for some variable P.
- **Effective**: can get answer in finite steps.
- System is **Decidable**: there is an effective procedure for establishing the truth of any formula.

Rules of Inference

$\alpha \dashv \beta$ or $$\frac{\alpha}{\beta}$$

- Modus Ponens
- And-Elimination
- Or-Introduction
- Double-Negation Elimination
- Unit Resolution
- Resolution

Complexity

- Determination of satisfiability of an arbitrary problem is a key hard problem. It is in the class of **NP-complete** problems.
- **Except:**
  - **For a formula in CNF, if each disjunct has only 2 literals, we can "efficiently" determine satisfiability.**
- Note: A is **valid** only if *not(A)* is not satisfiable.
  - Thus, validity is a hard question too.
-

Automated Theorem Proving

- Assume proper axioms of the form

$$(P_1 \wedge P_2 \wedge \dots P_n) \Rightarrow Q$$

- A **fact** is a propositional variable this is given.

- If we want to prove goal Q, we can do that by proving $(P_1 \wedge P_2 \wedge \dots P_n)$.
  - Q is **reduced to** $(P_1 \wedge P_2 \wedge \dots P_n)$.

- **ATP**: recursively try to reduce the sentences (goals) to be proven to a the facts we started with.

Predicate Calculus

- Also known as **first order logic**.

- A formal system with a "world" made up of
  - Objects
  - Properties of objects
  - Relations between objects.
  - Adds **quantification** <u>over objects</u> to propositional logic.
    - Note: <u>second order</u> logic includes quantification over classes.

$$\forall \mathbf{x}\ \ \mathbf{passes\_final(x)} \Rightarrow \mathbf{gets\_credit(x)}$$
$$\exists \mathbf{x}\ \ \mathbf{passes\_final(x)} \Rightarrow \mathbf{gets\_credit(x)}$$

FOL components

- Relations can be functions

```
Hair_color_of()
Is_student()
Took_ai424()
```

But they don't have to be

```
Son_of()
Owns_CD_titled()
```

FOL terminology

- **Terms**: represent objects, can be constants or expressions.
- **Predicate symbols**: a relation (sometimes functional).
- **Sentences**: as with propositional logic
- **Arity**: number of arguments to a relation
- **Atomic sentence**: predicate symbols and terms

  ```
  Owns_printer_model(brother_of(Sue),HP_DJ550)
  ```
- **Scope** of a quantifier: part of formula a quantifier applies to.

Complexity of ATP in FOL?

- First order logic is <u>universal</u>.
  - Any inference or computation we know of can be described.
    - We can describe the operation of a Turing machines.
  - Thus, **entailment is semidecidable**.
    - We can't tell if a computation halts except by running it an waiting… maybe forever.
- Much effort on restricting FOL to assure it is decidable.
  - It still may be "exponentially difficult".