Chapter 9

Knowledge: Representation

Introduction to Artificial Intelligence

Chandra Gummaluru
University of Toronto

Version W22.3

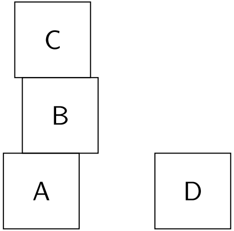
- The following is based on material developed by many individuals, including (but not limited to):

  - Sheila McIlraith
  - Bahar Aameri
  - Fahiem Bacchus
  - Sonya Allin

- Part of being an intelligent agent involves being able to infer implicit facts based on known or assumed ones.

  Example: Stacking Blocks

  - Suppose four blocks are arranged as follows:
    - block A is below block B
    - block B is below block C
    - block D is below block C
  - Is block A below block C?



- Humans develop this ability through experience. Our goal is to instil artificial agents with the same ability.

- Without reasoning, we would have to explicitly remember every fact we've learned.

- To achieve this ability, we do two things:

  1. **Represent** (encode) known statements in our brain.
  2. **Reason** (infer) new statements from the known ones.

- Thus, to achieve this ability artificially, we need to do things:

  1. Develop a **formal languages** to represent statements (this chapter).
  2. Develop a **reasoning mechanism** for the formal system (next chapter).

- There are many formal languages and reasoning mechanisms we could use.
- We will consider a representation called **first-order logic** (FOL), and a reasoning mechanism called **resolution**.

- Before studying any specifics, it is worth considering what the purpose of formal language is.
- Roughly speaking, the intent is to develop a "language" of sorts but with strict rules to avoid any ambiguity.
- **Example**: Ambiguity in English
  - What is the appropriate response to the request, "call me an ambulance,"?
    - "okay."
    - "uh...you're an ambulance...".
- The ambiguity arises from the fact that, in English (and other languages), there are multiple interpretations of many words/phrases.

- To avoid such ambiguity, a formal language must define the notion used to build its statements, as well as a system for interpreting those statements.
- The notion is called the **syntax** and the interpretations are the **semantics**.
- So, a formal language needs to provide syntax and a way to introduce semantics. However, it does not provide the semantics themselves.

- To clarify the difference between syntax and semantics, let us first consider a simpler formal language called **propositional logic** (PL).

- PL syntax consists of the following components:

  1. **binary variables**, where each variable is, by definition, a **formula**.
     - E.g., x.

- A PL **vocabulary** is set, $\mathcal{V}$ of (binary) variables.

- For any $v \in \mathcal{V}$ variable, the expression v is called an atomic $\mathcal{V}$-formula.
- Non-atomic $\mathcal{V}$-formulae are defined recursively as follows:
  - **negation**: $\neg f$, where f is any $\mathcal{V}$-formula.
  - **disjunction**: $f_1 \vee f_2$, where f_1 and f_2 are $\mathcal{V}$-formulae.
  - **conjunction**: $f_1 \wedge f_2$, where f_1 and f_2 are $\mathcal{V}$-formulae.
  - **implication**: $f_1 \rightarrow f_2$, where f_1 and f_2 are $\mathcal{V}$-formulae.

- The semantics for PL variables come from a **truth assigner**, $\mathcal{T} : \mathcal{V} \to \{\top, \bot\}$.
- We define an **extended truth assigner**, $\tilde{\tau}$, for all $\mathcal{V}$-formulae, f, f_1, f_2, as follows:
  - $\tilde{\tau}(x) = \tau(x)$, for any $x \in \mathcal{V}$
  - $\tilde{\tau}(\neg f) = \top$ iff $\tau(f) = \bot$
  - $\tilde{\tau}(f_1 \vee f_2) = \top$ iff $\tilde{\tau}(f_1) = \top$ or $\tilde{\tau}(f_2) = \top$
  - $\tilde{\tau}(f_1 \wedge f_2) = \top$ iff $\tilde{\tau}(f_1) = \top$ and $\tilde{\tau}(f_2) = \top$
  - $\tilde{\tau}(f_1 \to f_2) = \top$ iff $\tilde{\tau}(\neg f_1) = \top$ or $\tilde{\tau}(f_2) = \top$.

- FOL syntax consists of the following components:

  1. **variables**, where each variable is, by definition, a **term**.
     - E.g., x.
  2. **functions**, which each map many terms to a single term.
     - E.g., below(x), which returns the block directly below x.
  3. **predicates**, which each map many terms to true/false.
     - E.g., isBelow(x, y), which returns whether y is below x or not.

- An FOL **vocabulary** is a triple, $\mathcal{L} = (\mathcal{V}, \mathcal{F}, \mathcal{P})$, where $\mathcal{V}, \mathcal{F}$, and $\mathcal{P}$ are sets of variables, functions, and predicates, respectively.

- Let $\mathcal{L}$ be a vocabulary.
- For any n-ary $\mathcal{L}$-predicate P, and $\mathcal{L}$-terms, $t_1, \ldots, t_n$, the expression, $P(t_1, \ldots, t_n)$ is called an atomic $\mathcal{L}$-formula.
  - E.g., $\mathrm{isAbove}(x, \mathrm{below}(y))$.
- Atomic formula represents the most fundamental statements.

- Non-atomic $\mathcal{L}$-formulae are defined recursively as follows:
  - **negation**: $\neg f$, where f is any $\mathcal{L}$-formula.
  - **disjunction**: $f_1 \vee f_2$, where f_1 and f_2 are $\mathcal{L}$-formulae.
  - **conjunction**: $f_1 \wedge f_2$, where f_1 and f_2 are $\mathcal{L}$-formulae.
  - **implication**: $f_1 \rightarrow f_2$, where f_1 and f_2 are $\mathcal{L}$-formulae.
  - **existential**: $\exists x f$, where x is a variable and f is any $\mathcal{L}$-formula.
  - **universal**: $\forall x f$, where x is a variable and f is any $\mathcal{L}$-formula.

- In FOL, the semantics are provided by what we refer to as a **model**. The model, $\mathcal{M}$, consists of the following components:

1. a **domain of discourse**, M, which is a set of relevant elementary objects.
   - E.g., $M = \{A, B, C, D\}$, representing the blocks.
2. **specializations of functions**, $f^{\mathcal{M}} : M^n \rightarrow M$, for each n-ary function, f, so that $f^{\mathcal{M}}$ assigns f for the domain of discourse.
   - E.g., $\text{above}^{\mathcal{M}}(A) = B$, $\text{above}^{\mathcal{M}}(B) = C$, $\text{above}^{\mathcal{M}}(C) = C$, $\text{above}^{\mathcal{M}}(D) = D$.
3. **specializations of predicates**, $p^{\mathcal{M}} \subseteq M^n$, for each n-ary predicate, p, so that $p(t_1, \ldots, t_n)$ is true if and only if $(t_1, \ldots, t_n) \in p^{\mathcal{M}}$.
   - E.g., $\text{isBelow}^{\mathcal{M}} = \{(B, A), (C, B), (B, D)\}$.

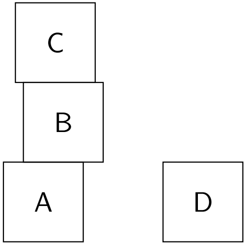
- In PL, variables are meant to represented Boolean expressions.
- In FOL, variables (and terms in general), are meant to represent objects in a universe defined by some model, $\mathcal{M}$:
  - atomic formulae represent fundamental properties and relations that hold about those elements.
  - other formulae represent complex assertions whose truth values depend on the atomic formulae within them.

- To do this, we need to bind the variables in $\mathcal{V}$, with elements in the domain of discourse, M:
  - We define an **assignment function**, $\sigma : \mathcal{V} \to M$, so that $\sigma(x)$ is the element in the universe represented by the variable x.
  - To bind any $\mathcal{L}$-terms, we recursively define an **extended assignment function**, $\bar{\sigma}$ so that $\bar{\sigma}(x) = \sigma(x)$ and $\bar{\sigma}(f(t_1, \ldots, t_n)) = f^{\mathcal{M}}(\bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_n))$.

Example: Stacking Blocks

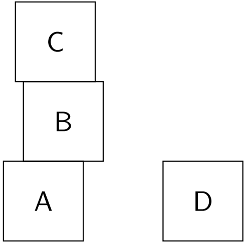
- Suppose we have a vocabulary, $\mathcal{L}$, with the functions
  - below(x), the block directly below x (or x if none)
  - above(x), the block directly above x (or x if none)

  and predicates
  - isBelow(x, y), y is below x
  - isAbove(x, y), y is above x



Example: Stacking Blocks

- A model, $\mathcal{M}$, for the situation shown is:
  - $M = \{A, B, C, D\}$.
  - isBelow$^{\mathcal{M}}$ $= \{\langle B, A\rangle, \langle C, B\rangle, \langle B, D\rangle\}$
  - isAbove$^{\mathcal{M}}$ $= \{\langle A, B\rangle, \langle B, C\rangle, \langle D, B\}$
  - below$^{\mathcal{M}}(A) = A$, below$^{\mathcal{M}}(B) = A$, below$^{\mathcal{M}}(C) = B$, below$^{\mathcal{M}}(D) = D$
  - above$^{\mathcal{M}}(A) = B$, above$^{\mathcal{M}}(B) = C$, above$^{\mathcal{M}}(C) = C$, above$^{\mathcal{M}}(D) = D$
- Suppose we let
  - $\mathcal{V} = \{v_1, \ldots, v_4\}$
  - $\sigma(v_1) = D, \sigma(v_2) = C, \sigma(v_3) = B, \sigma(v_4) = A$



Example: Stacking Blocks

- We can compute the value of an $\mathcal{L}$-term like below(below(v_2)) as:

$$\bar{\sigma}\,(\text{below}(\text{below}(v_2))) = \text{below}^{\mathcal{M}}(\bar{\sigma}(\text{below}(v_2)))$$

$$= \text{below}^{\mathcal{M}}\left(\text{below}^{\mathcal{M}}(\bar{\sigma}(v_2))\right)$$

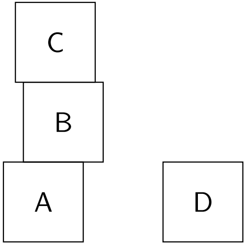
$$= \text{below}^{\mathcal{M}}\left(\text{below}^{\mathcal{M}}(\sigma(v_2))\right)$$

$$= \text{below}^{\mathcal{M}}\left(\text{below}^{\mathcal{M}}(C)\right)$$

$$= \text{below}^{\mathcal{M}}(B)$$

$$= A$$

Notice that the value depends on both $\mathcal{M}$ and σ.



- We write $\mathcal{M} \vDash f[\sigma]$ to denote that $\mathcal{M}$ **satisfies** the formula, f, under σ. It is defined recursively as follows:
    - $\mathcal{M} \vDash P(t_1, \ldots, t_n)[\sigma]$ if and only if $\langle \bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_n) \rangle \in p^{\mathcal{M}}$
    - $\mathcal{M} \vDash (t_1 = t_2)[\sigma]$ if and only if $\bar{\sigma}(t_1) = \bar{\sigma}(t_2)$
    - $\mathcal{M} \vDash \neg f[\sigma]$ if and only if $\mathcal{M} \nvDash f[\sigma]$
    - $\mathcal{M} \vDash (f_1 \vee f_2)[\sigma]$ if and only if $\mathcal{M} \vDash f_1[\sigma]$ or $\mathcal{M} \vDash f_2[\sigma]$
    - $\mathcal{M} \vDash (f_1 \wedge f_2)[\sigma]$ if and only if $\mathcal{M} \vDash f_1[\sigma]$ and $\mathcal{M} \vDash f_2[\sigma]$
    - $\mathcal{M} \vDash (\forall x f)[\sigma]$ if and only if $\mathcal{M} \vDash f[\sigma[x, m]]$ for all $m \in M$
    - $\mathcal{M} \vDash (\exists x f)[\sigma]$ if and only if $\mathcal{M} \vDash f[\sigma[x, m]]$ for some $m \in M$
- Here $\sigma[x, m]$ is defined assuming x is fixed and so that

$$\sigma[x, m](y) = \begin{cases} \sigma(y), y \neq x \\ m, y = x \end{cases}$$

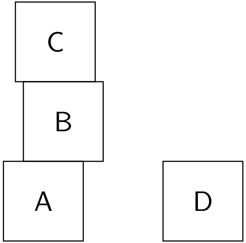
Example: Stacking Blocks

- We can determine whether $\mathcal{M}$ satisfies an $\mathcal{L}$-formula like $\exists v \text{isAbove}\,(v, \text{below}(\text{below}(v_2)))$ under σ by checking whether $\mathcal{M}$ satisfies it under $\sigma[v, m]$ for some $m \in M$:

$$\text{isAbove}^{\mathcal{M}}\,(\bar{\sigma}[v, m](v), \bar{\sigma}(\text{below}(\text{below}(v_2))))$$

$$= \text{isAbove}^{\mathcal{M}}\,(m, A)$$

There is no $m \in M$ such that $\langle m, A\rangle \in \text{isAbove}^{\mathcal{M}}$ and so the original statement is not satisfied by $\mathcal{M}$ under σ.



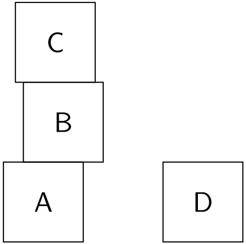
- An occurrence of a variable, x, in an FOL formula, f, is **bound** if and only if it is in a sub-formula of the form $\forall x f'$ or $\exists x f'$. Otherwise, x is **free**.
- Bound variables cannot be assigned fixed values and thus,the validity of formulae containing only bound variables is independent of σ.
- Such formulae are called **sentences**.
- Formally, for any sentence, s, and assignments σ and σ', we have

$$\mathcal{M} \vDash s[\sigma] \text{ if and only if } \mathcal{M} \vDash S[\sigma'].$$

- Thus, for any sentence, s, we simply write $\mathcal{M} \vDash s$ to denote that $\mathcal{M}$ satisfies s.

Example: Stacking Blocks

- We saw that a model, $\mathcal{M}$, for the situation shown is:
  - $M = \{A, B, C, D\}$.
  - $\text{isBelow}^{\mathcal{M}} = \{\langle B, A\rangle, \langle C, B\rangle, \langle B, D\rangle\}$
  - $\text{isAbove}^{\mathcal{M}} = \{\langle A, B\rangle, \langle B, C\rangle, \langle D, B\rangle\}$
  - $\text{below}^{\mathcal{M}}(A) = A$, $\text{below}^{\mathcal{M}}(B) = A$, $\text{below}^{\mathcal{M}}(C) = B$, $\text{below}^{\mathcal{M}}(D) = D$
  - $\text{above}^{\mathcal{M}}(A) = B, \text{above}^{\mathcal{M}}(B) = C, \text{above}^{\mathcal{M}}(C) = C, \text{above}^{\mathcal{M}}(D) = D$
- The sentence, $\forall x \forall y\, (\text{isBelow}(x, y) \rightarrow \text{isAbove}(y, x))$ is satisfied by $\mathcal{M}$.



- Let Φ be a set of sentences.
- We say $\mathcal{M}$ satisfies Φ, denoted $\mathcal{M} \vDash \Phi$ iff $\mathcal{M} \vDash s$ for every sentence, $s \in \Phi$.
- We say that Φ is **satisfiable** if there exists some model $\mathcal{M}$ such that $\mathcal{M} \vDash \Phi$.