ME 620: Fundamentals of Artificial Intelligence

Lecture 18: Inference in FOL - Part I

Shyamanta M Hazarika
Biomimetic Robotics and Artificial Intelligence Lab
Mechanical Engineering and M F School of Data Sc. & AI
IIT Guwahati

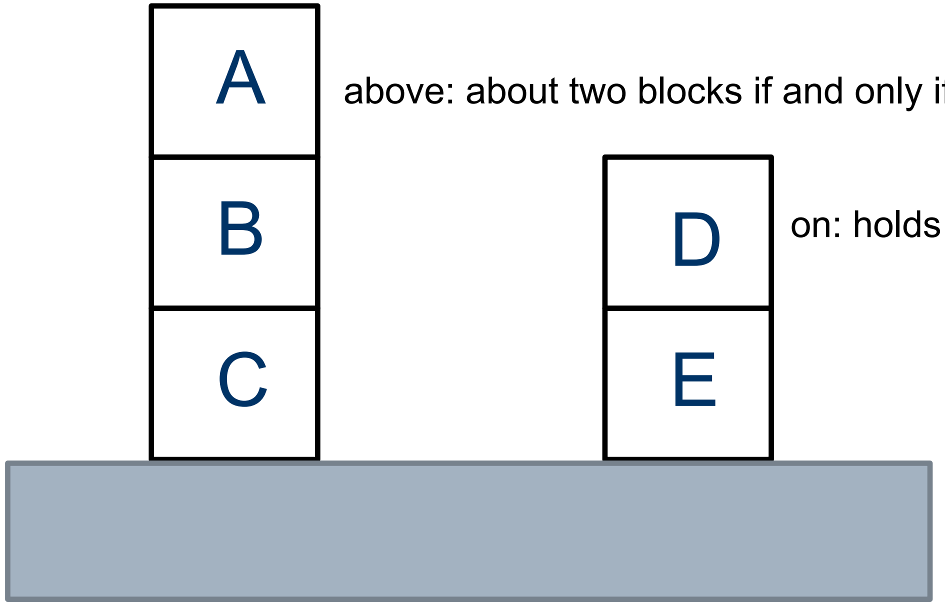
Conceptualization

Blocks World

Elements of a conceptualization: objects; functions and relations

Formally, a **conceptualization** is a triple consisting of a universe of discourse, a functional basis set for that universe of discourse, and a relational basis set.

clear: to mean no block is on top of the block



above: about two blocks if and only if one is above the other.

on: holds if and only if one is immediately above the other.

The following is one conceptualization of the BLOCKS world here,

⟨{A, B, C, D, E},
{hat},
{on, above, clear, table}⟩

table: to mean a block is on the table

BLOCKS WORLD scene

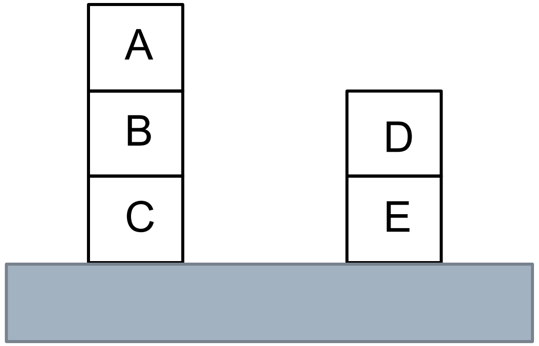
Interpretation

Blocks World

Interpretation *I* is a mapping between elements of the language and elements of a conceptualization

FOPC language has the five object constants: A, B, C, D, AND E.

The following mapping correspond to our usual interpretation for these symbols.



BLOCKS WORLD scene

$A^I = A$

$B^I = B$

$C^I = C$

$D^I = D$

$E^I = E$

The function constant hat is mapped to the tuples corresponding to this function.

Relation constants are mapped to the each of their extension.

$hat^I = \{\langle B, A \rangle, \langle C, B \rangle, \langle E, D \rangle\}$

$on^I = \{\langle A, B \rangle, \langle B, C \rangle, \langle D, E \rangle\}$

$above^I = \{\langle A, B \rangle, \langle B, C \rangle, \langle A, C \rangle, \langle D, E \rangle\}$

$table^I = \{C, E\}.$

$clear^I = \{D, A\}.$

Knowledge Representation

Blocks World Example

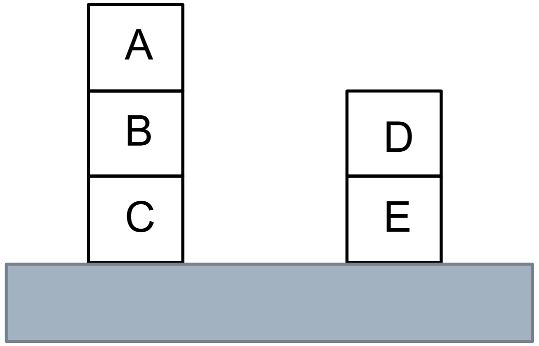
Essential Information

on(A,B) above(A,B) clear(A)

on(B,C) above(B,C) clear(D)

on(D,E) above(A,C) table(C)

above(D,E) table(E)



BLOCKS WORLD scene

Encode some general facts.

General Sentences

$\forall x\ \forall y\ (on(x,y) \rightarrow above(x,y))$

$\forall x\ \forall y\ \forall z\ (above(x,y) \wedge above(y,z) \rightarrow above(x,z))$

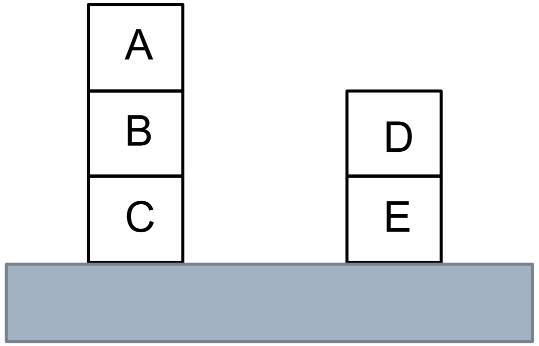
$\forall x\ (clear(x) \rightarrow \neg\exists y\ on(y,x))$

These general statements ALSO apply to Blocks World scenes other than the one pictured here.

Knowledge Representation

Blocks World Example

Given the general sentences and the ON relations; we may not have explicitly the ABOVE relations.



BLOCKS WORLD scene

Essential Information

on(A,B) clear(A)

on(B,C) clear(D)

on(D,E) table(C)

table(E)

General Sentences

A conjunction of these formulas can serve as a description of the `world state'.

$$\forall x\ \forall y\ (on(x,y) \rightarrow above(x,y))$$

$$\forall x\ \forall y\ \forall z\ (above(x,y) \wedge above(y,z) \rightarrow above(x,z))$$

$$\forall x\ (clear(x) \rightarrow \neg \exists y\ on(y,x))$$

Suppose the problem is to show that a certain property is true in a given state. For example, we might want to establish that there is nothing on block A.

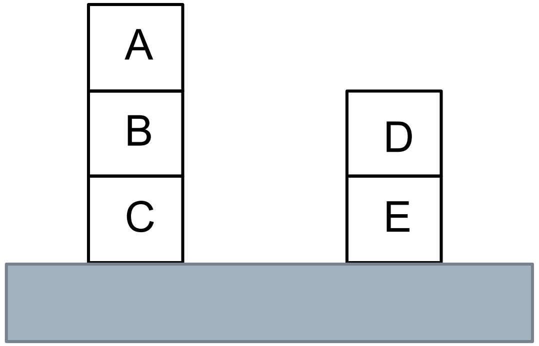
Knowledge Representation

Blocks World Example

$\neg\exists y \; on(y,A)$

Essential Information

on(A,B)	clear(A)
on(B,C)	clear(D)
on(D,E)	table(C)
	table(E)



BLOCKS WORLD scene

General Sentences

A conjunction of these formulas can serve as a description of the `world state'.

$$\forall x \; \forall y \; (on(x,y) \rightarrow above(x,y))$$

$$\forall x \; \forall y \; \forall z \; (above(x,y) \wedge above(y,z) \rightarrow above(x,z))$$

$$\forall x \; (clear(x) \rightarrow \neg\exists y \; on(y,x))$$

We can DEDUCE this FACT by showing that the formula logically follows from the state description; equivalently the formula could be derived from the state description by application of SOUND rules of inference.

Making Inferences using FOPC

- There exists well-understood mechanisms for making inferences from predicate-calculus well-formed formulas.
  - The terminology used in discussing this is the terminology of mathematical proof.

1. An axiom is a well-formed formula that is asserted to be true without proof.

   In an AI system, the axioms would be:
   - The domain-specific knowledge rules in the database, and
   - The input data supplied by the user.

Making Inferences using FOPC

2. A theorem is a well-formed formula that can be proven true on the basis of the axioms.

In an AI system, the theorems would be:

- Inferences that can be drawn from the rules and input data (in a forward chaining system.)
- Questions posed by the user.
  - Note, that a question can be posed as a theorem!
  - "Who chases Jerry?" can be turned into a predicate calculus theorem: $\exists x(chases(x, Jerry))$

Method of proof used with theorems containing existentially quantified variables has, as a side effect, the finding in the knowledge base of a value for the variable for which the desired condition holds.

Making Inferences using FOPC

3. “Reasoning” in logic-based AI system is accomplished by using methods of mathematical proof.

- Since these have a long history, they provide a wealth of resources for us to draw on in doing AI.
- One of our most important tools are the laws of inference which allow us to form new theorems from axioms and other theorems.
    - Derived well formed formula are the theorems; sequence of inference rule applications used in the derivation constitute a Proof of the theorem.

Rules of Inference

In formulating proofs, one of our most important tools are the Laws of Inference

1. Modus ponens

$$\frac{A \rightarrow B, A}{B}$$

A: It is snowing outside
B: It is cold outside

Premises
$A \rightarrow B$: It is snowing outside implies it is cold outside.
A : It is snowing outside

Conclusion*
B : It is cold outside.

* Reasonable to infer

Rules of Inference

1. Modus ponens

$$\frac{A \rightarrow B, A}{B}$$

A: It is snowing outside
B: It is cold outside

2. Modus tolens

$$\frac{A \rightarrow B, \neg B}{\neg A}$$

Premises

$A \rightarrow B$: It is snowing outside implies it is cold outside.
$\neg B$: It is not cold outside

Conclusion*

$\neg A$: It is not snowing outside.

* Reasonable to infer

Rules of Inference

1. Modus ponens

$$\frac{A \rightarrow B, A}{B}$$

A: It is snowing outside
B: It is cold outside

2. Modus tolens

$$\frac{A \rightarrow B, \neg B}{\neg A}$$

Premises

$A \rightarrow B$: It is snowing outside implies it is cold outside.
B : It is cold outside

NOTE that it is NOT SOUND to say
A : It is snowing outside.

If we are told it is cold outside; it is NOT NECESSARILY the case that it is snowing outside.

This sort of reasoning is called Abduction.

Rules of Inference

1. Modus ponens

$$\frac{A \rightarrow B, A}{B}$$

A $\quad$ $\neg A \vee B$

B

2. Modus tolens

$$\frac{A \rightarrow B, \neg B}{\neg A}$$

$\neg A \vee B$ $\quad$ $\neg B$

$\neg A$

3. Resolution

$$\frac{A \vee B, \neg B \vee C}{A \vee C}$$

$A \vee B$ $\quad$ $\neg B \vee C$

$A \vee C$

Rules of Inference

- Rules of Inference introduced in Propositional Logic can be also used in Predicate Logic
  - One would need to learn **how to deal with formulas that contain variables**.

1. Universal Specialization – Universal Instantiation
2. Existential Instantiation
3. Existential Generalization
4. Universal Generalization - Universal Introduction

Universal Specialization

- $$\frac{\forall x\ P(x)}{P(C)}$$
  where C is *any* constant symbol.

Universal Specialization is also referred to as Universal Instantiation.

- Example:
  - $\forall x\ eats(Zen, x) \rightarrow eats(Zen, IceCream)$

The **variable symbol can be replaced by any ground term**, i.e., any constant symbol or function symbol applied to ground terms only.

Existential Instantiation

- $$\frac{\exists x\ P(x)}{P(A)}$$

  Where A is a *brand-new* constant symbol.

- Example:
  - $\exists x$ likes(Zen, x) $\rightarrow$ likes(Zen, Stuff)

  Note that the **variable is replaced by a brand-new constant** not occurring in this or any other sentence in the KB.

Existential Instantiation

- $$\frac{\exists x\ P(x)}{P(A)}$$

  Where A is a *brand-new* constant symbol.

- Example:
  - $\exists x\ likes(Zen, x) \rightarrow likes(Zen, Stuff)$

Also known as skolemization; constant is a **skolem constant**. Convenient to reason about the unknown object, rather than the existential quantifier.

Existential Generalization

- $$\frac{P(c)}{\exists x\, P(x)}$$

- Example
  - eats(Zen, IceCream) → ∃x eats(Zen, x)

All instances of the given **constant symbol are replaced by the new variable symbol**. Note that the variable symbol cannot already exist anywhere in the expression.

Universal Generalization

- $\dfrac{P(c)}{\forall x\, P(x)}$

  If P(c) must be true, and we have assumed nothing about c, then ∀ x P(x) is true.

  Universal generalization is the rule of inference that states that ∀ xP(x) is true, given the premise that P(c) is true for all elements c in the domain.

  Universal generalization is used when we show that ∀ xP(x) is true by taking an arbitrary element c from the domain and showing that P(c) is true.

Rules of Inference, Theorems and Proofs

- Rules of inference can be applied to well-formed formulas to produce new well-formed formulas.
  - Derived well-formed formulas are referred to as Theorems.
  - Sequence of inference rule application used in the derivation constitutes the proof of the theorem.
- For proving theorems involving quantified formulas, it is often **necessary to *match* certain subexpressions**.

Unification

- Example

$$\neg W1(x) \lor W2(x) \qquad \neg W1(A)$$

$$W2(A)$$

1. $\forall x$ [$W1(x) \rightarrow W2(x)$]
2. $W1(A)$

For *universal specialization* to produce W2(A) from 1 and 2 above; it is necessary to find the substitution A/x.

- Finding **substitutions of terms for variables** to make expressions identical is an extremely important process and is called **unification.**
- The **set of substitutions is called a unifier**.

Unification

- Example

$\neg W1(x) \lor W2(x)$ $\neg W1(A)$

$W2(A)$

1. $\forall x\ [\ W1(x) \rightarrow W2(x)]$
2. $W1(A)$

For *universal specialization* to produce W2(A) from 1 and 2 above; it is necessary to find the substitution A/x.

- Unification makes **resolution of clauses containing variables** possible.
- Unifier(s) used in a resolution proof **provide a handle for using the proof outcome to answer questions**.

Unification

- The terms of an expression can be variable symbols, constant symbols or functional expressions, the latter consisting of function symbols and terms.

- A **substitution instance** of an expression is obtained by substituting terms for variables in that expression.

Example: Four instances of substitution of P[x, f(y), B].

P[z, f(w), B]	Alphabetic variant
P[x, f(A), B]	
P[g(z),f(A),B]	
P[C, f(A), B]	Ground Instance

The last of the four instances shown is called a ground instance, since none of the terms in the literal contains variables.

Unification

- For two formulas Φ and Ψ; at least one of which contain variables, there is a **substitution U that makes them identical**. U is a unifier for Φ and Ψ.

  Example: P(A, x,) and P(y,z)

  U = {A/y,x/z}

  Often we have more than one unifier for a pair of formulas.

  Example: U = {A/y,B/x, B/z} is another unifier above.

- Variables or term containing variables can also be used for another variable.

Unification

- Necessary **unifier** will be apparent when examining two clauses. There is a whole body of theory on unification.

Including a unification algorithm; which we shall not cover here.

- A **unification is a substitution**; is a set of pairs $\{t_1/V_1, t_2/V_2, t_3/V_3 \ldots\}$ meaning t_1 is to be substituted for V_1, t_2 for V_2 ….

- We write an **expression E followed by a substitution s** to denote the instance of expression E that results from making the substitution s

Example $P(A, phi(A,B)) \leftrightarrow [P(x, phi(x,y))]\{A/x,B/y\}$

Unification

- The **composition of two substitutions** s1 and s2 is denoted as s1s2, is a substitution obtained by applying s2 to terms of s1; and adding any pairs of s2 having variables not occurring among the variables of s1.

  Example: s1={g(x,y)/z} and s2={A/x, B/y, C/w, D/z}

  s1s2 = {g(A,B)/z, A/x, B/y,C/w}

- Properties
  - (Es1)s2 = E(s1s2)
  - Composition of substitution is associative

    (s1s2)s3 = s1(s2s3)
  - Substitutions are not in general commutative

    s1s2 $\neq$ s2s1

Most General Unifier

- When there exist multiple possible unifiers for an expression E, there is at least one, called the **most general unifier, mgu, g** of E, that has the property that if s is any unifier for E yielding Es, then there exist a substitution s′ such that Es = Egs′

Example: P(A, x,) and P(y,z);

g = {A/y,x/z} is an mgu

For s′ = {B/x}, we get

s = {A/y,B/x, B/z}

If we apply mgu, g and then apply the second substitution s’, we get s. Note that the reverse would not be possible.

Most General Unifier

- The **mgu preserves as much generality as possible** for a pair of formulas; by using the mgu we **leave maximum flexibility for the resolvent** to resolve with other clauses.

- The **most general unifier is not necessarily unique**.

  Example P(A, x,) and P(y,z);

  {A/y, z/x} is also an mgu.

There are many algorithms that can be used to unify a finite set of unifiable expressions.

Clauses

A formula is in conjunctive normal form (CNF) or clausal normal form if it is a conjunction of one or more clauses, where a clause is a disjunction of literals

- A **clause** is defined as a well-formed formula consisting of a disjunction of literals.

  Example

  1. $P1 \vee P2 \vee P3 \ldots. \vee PN$
  2. $\neg Q1(x1) \vee \neg Q2(fn(x2))$

- Any predicate calculus well-formed formula can be converted to a set of clauses.

Before we focus on the process of Resolution and proofs through Resolution Refutation, we discuss in the remainder of the Lecture today, how any FOPC well-formed formula can be converted to a set of clauses.

Converting to Clausal Form

Step – I : Eliminate Implication Symbols

Example $\forall x\,[\,W1(x) \rightarrow [\forall y\,[W2(y) \rightarrow W3(f(x,y))]]]$

$\forall x\,[\neg W1(x) \vee [\forall y\,[\neg W2(y) \vee W3(f(x,y))]]]$

All occurrences of the $\rightarrow$ symbol in a well-formed formula are eliminated by making the substitution

$[\neg X \vee Y]$ for $[X \rightarrow Y]$

Step – II : Reduce scopes of Negation Symbols

Example $\neg\forall y\,[Q(x,y) \rightarrow P(y)]$

$\exists y\,[Q(x,y) \wedge \neg P(y)]$

$\exists y\,\neg\,[Q(x,y) \rightarrow P(y)]$
$\exists y\,\neg\,[\neg\,Q(x,y) \vee P(y)]$
$\exists y\,[Q(x,y) \wedge \neg\,P(y)]$

We want each negation symbol to apply to at most one atomic formula. Achieve this by repeated use of De Morgan's Laws and other equivalences.

Converting to Clausal Form

Step – III : Standardize variables

Example $\forall x\ [\ W1(x) \rightarrow \exists x\ W2(x)]$

$\forall x\ [\ W1(x) \rightarrow \exists y\ W2(y)]$

The scope of a variable is the sentence to which the quantifier syntactically applies.

Within the scope of any quantifier, a variable bound by the quantifier is a dummy variable. It can be uniformly replaced by any other (non-occurring) variable throughout the scope of the quantifier without changing the truth value of the well-formed formula.

Standardizing variable refers to renaming the dummy variables to ensure that each quantifier has its own unique dummy variable.

Converting to Clausal Form

Step – IV : Eliminate Existential Quantifiers

Example 1. $\forall y\ [\exists x\ P(x,y)]$

$\forall y\ [P(g(y),y)]$

Using the Skolem function in place of x that exists, we can eliminate the existential quantifier altogether and write the universally quantified sentence.

In Example 1. for all y, there exists x (possibly depending on y) such that P(x,y) is true. Note that the existential quantifier is within the scope of the universal quantifier. We allow the possibility that the x depends on the value of y.

Explicitly defined by function g(y); which maps each value of y into x that `exists'. Such a function is called a **Skolem function**.

Converting to Clausal Form

Step – IV : Eliminate Existential Quantifiers

Example 1. $\forall y\ [\exists x\ P(x,y)]$

$\forall y\ [P(g(y),y)]$

2. $\exists x\ P(x)$

$P(A)$

In Example 2 the existential quantifier being eliminated is not within the scope of the universal quantifier. We use a Skolem function of no arguments.

Explicitly state a constant A, used to refer to the entity that we know `exists’. Such a constant is called a **Skolem constant**.

It is important that A be a new constant symbol; one not used in other formulas to refer to known entities.

Converting to Clausal Form

Step – V : Convert to Prenex Form

There are no remaining existential quantifier; Each Universal quantifier has its own variable.

Move all universal quantifiers to front of well-formed formula; scope of each quantifier is the entirety of the formula.

The resulting well-formed formula is in **prenex form**.

The prenex form consists of a **string of quantifiers called prefix** followed by a quantifier-free formula called the matrix.

$$\forall x\ \forall y\ \forall z\ \forall w \ \ldots\ldots\ [P(x,y)Q(g(z),y)R(w)\ \ldots\ldots]$$

Converting to Clausal Form

Step – VI : Put in Conjunctive Normal Form

Example $P \vee (Q \wedge R)$

Conjunction of a finite set of disjunctions of literals

$(P \vee Q) \wedge (P \vee R)$

Any matrix may be written as the **conjunction of a finite set of disjunction of literals**. Such a matrix is said to be in **conjunctive normal form**.

Recall that a quantifier-free formula called the matrix.

May put any matrix into a conjunctive normal form by repeatedly using one of the distributive rules as highlighted above.

Converting to Clausal Form

Step – VII : Eliminate Universal Quantifiers

All variables remaining at this stage are universally quantified; bound. Eliminate the explicit reference.

Left with a matrix in Conjunctive Normal Form.

Step – VIII : Eliminate ∧ Symbols

Example $P \wedge (Q \vee R)$

1. P
2. $Q \vee R$

Eliminate the explicit reference of AND. Result of repeated replacement is to obtain a finite set of well-formed formula, each of which is a disjunction of literals.

Step – IX : Rename variables

Variables symbols may be renamed so that no variable symbol appears in more than one clause; Standardizing variables apart.

Resolution

- First step for using **resolution as a rule of inference** is to get the formulas converted into clauses.
  - If a well-formed formula Φ logically follows from a set of well-formed formulas S, then it also logically follows from the set of clauses obtained by converting the well-formed formulas in S to clause form.
  - Clauses are a completely general form in which to express the well-formed formulas.
- Iteratively **applying the resolution rule in a suitable way** allows for proving that a first-order formula is unsatisfiable.
  - **Resolution Refutation Systems** allow proving a theorem by adding its negation to the clauses; and arriving at a contradiction.