CS 161 Fundamentals of Artificial Intelligence
Lecture 10

First-order Logic: Inference

Quanquan Gu

Department of Computer Science
UCLA

March 2, 2023

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Resolution

Universal Instantiation (UI)

- Whenever a KB contains a universally quantified sentence, we may add to the KB any instantiation of that sentence, where the logic variable v is replaced by a concrete ground term g:
- For any variable v and ground term g, we denote substitution θ as $\theta = \{v/g\}$, $\textsc{Subst}(\theta, \alpha)$ as the result of applying the substitution θ to the sentence α.
- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v \;\; \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

E.g., $\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$ yields

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John))$$
$$\;\Rightarrow\; Evil(Father(John))$$

Existential Instantiation (EI)

- Whenever a KB contains an existentially quantified sentence $\exists v\ \alpha$, we may add a single instantiation of that sentence to the KB, where the logic variable v is replaced by a **Skolem constant** symbol k which must not appear elsewhere in the knowledge base:

$$\frac{\exists v\ \ \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

- Ex1, $\exists x\ \ Crown(x) \wedge OnHead(x, John)$ yields

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a new constant symbol, called a **Skolem constant**
- Ex2: from $\exists x\ \ d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided e is a new constant symbol

$\forall x, \exists y \quad Loves(x, y)$

~~$\forall x, \quad Loves(x, c)$~~

$\forall x, \quad Loves(x, c(x))$

skolem function

Reduction to propositional inference

- Instantiating all quantified sentences allows us to ground the KB, that is, to make the KB propositional
- Suppose the KB contains just the following:

$$\begin{array}{l} \forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x) \\ King(John) \\ Greedy(John) \\ Brother(Richard, John) \end{array}$$

- Instantiating the universal sentence in **all possible** ways, we have

1 $King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$

2 $King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$

3 $King(John)$ 4 $Greedy(John)$

5 $Brother(Richard, John)$

- The new KB is **propositionalized**: proposition symbols are

$King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$ etc.

Reduction contd.

- Every FOL KB can be propositionalized so as to preserve entailment
- Then, FOL inference can be done by: propositionalize KB and query, apply resolution, return result
- Problem: with function symbols, there are infinitely many ground terms $Father(Father(Father(John)))$

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a **finite** subset of the propositional KB

- For $n = 0$ to ∞ do
  - create a propositional KB by instantiating with depth-n terms
  - see if α is entailed by this KB
- Works if α is entailed, loops if α is not entailed!

Theorem: Turing (1936), Church (1936), entailment in FOL is **semidecidable** –that is, algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
E.g., from

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$\forall y \;\; Greedy(y)$$
$$Brother(Richard, John)$$

- It seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations
- With function symbols, it gets much worse!

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules can be applied.
- Unification: finding substitutions that make different logical expressions look identical. $\text{UNIFY}(\alpha, \beta) = \theta$ if $\text{SUBST}(\theta, \alpha) = \text{SUBST}(\theta, \beta)$
- Suppose we want to know: whom does John know?
- Answers to this query can be found by finding all sentences in the knowledge base that unify with $Knows(John, x)$.

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mom(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	
$Knows(John, x)$	$Knows(x_{17}, OJ)$	

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules can be applied.
- Unification: finding substitutions that make different logical expressions look identical. $\text{UNIFY}(\alpha, \beta) = \theta$ if $\text{SUBST}(\theta, \alpha) = \text{SUBST}(\theta, \beta)$
- Suppose we want to know: whom does John know?
- Answers to this query can be found by finding all sentences in the knowledge base that unify with $Knows(John, x)$.

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mom(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	
$Knows(John, x)$	$Knows(x_{17}, OJ)$	

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules can be applied.
- Unification: finding substitutions that make different logical expressions look identical. $\text{UNIFY}(\alpha, \beta) = \theta$ if $\text{SUBST}(\theta, \alpha) = \text{SUBST}(\theta, \beta)$
- Suppose we want to know: whom does John know?
- Answers to this query can be found by finding all sentences in the knowledge base that unify with $Knows(John, x)$.

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mom(y))$	$\{y/John, x/Mom(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	
$Knows(John, x)$	$Knows(x_{17}, OJ)$	

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules can be applied.
- Unification: finding substitutions that make different logical expressions look identical. $\textsc{Unify}(\alpha, \beta) = \theta$ if $\mathrm{SUBST}(\theta, \alpha) = \mathrm{SUBST}(\theta, \beta)$
- Suppose we want to know: whom does John know?
- Answers to this query can be found by finding all sentences in the knowledge base that unify with $Knows(John, x)$.

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mom(y))$	$\{y/John, x/Mom(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, OJ)$	

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules can be applied.
- Unification: finding substitutions that make different logical expressions look identical. $\textsc{Unify}(\alpha, \beta) = \theta$ if $\mathrm{SUBST}(\theta, \alpha) = \mathrm{SUBST}(\theta, \beta)$
- Suppose we want to know: whom does John know?
- Answers to this query can be found by finding all sentences in the knowledge base that unify with $Knows(John, x)$.

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mom(y))$	$\{y/John, x/Mom(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, OJ)$	$\{x/OJ, x_{17}/John\}$

Standardizing Apart

α	β	θ
$Knows(John, x)$	$Knows(x, OJ)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, OJ)$	$\{x/OJ, x_{17}/John\}$

- First unification fails: the problem arises only because the two sentences happen to use the same variable name x
- The problem can be avoided by standardizing apart one of the two sentences being unified, which means renaming its variables to avoid name clashes.

Standardizing apart eliminates overlap of variables, e.g., $Knows(x, OJ)$ to $Knows(x_{17}, OJ)$

Most General Unifier

- $\textsc{Unify}(\alpha, \beta)$ returns a substitution that makes α, β look the same
- Maybe more than one unifier! e.g., $\textsc{Unify}(Knows(John, x), Knows(y, z))$ could return $\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$; first one more general!
- For every unifiable pair of sentences, there is a single most general unifier (MGU) that is unique up to renaming and substitution of variables.
- For $\textsc{Unify}(Knows(John, x), Knows(y, z))$, the MGU is $\{y/John, x/z\}$.

Generalized Modus Ponens (GMP)

For atomic sentences p_i, p_i', q, where there is a substitution θ such that $\forall i, \mathsf{SUBST}(\theta, p_i') = \mathsf{SUBST}(\theta, p_i)$, then

$$\frac{p_1{}', \;\; p_2{}', \; \ldots, \; p_n{}', \;\; (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\mathsf{SUBST}(\theta, q)}$$

For example,

$p_1{}'$ is $King(John)$	p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$\mathsf{SUBST}(\theta, q)$ is $Evil(John)$	

GMP: definite clauses

- GMP used with KB of **definite clauses**:
- Definite clauses:
  - Atomic sentences
  - Implication whose premises is a conjunction of positive literals and whose conclusions is a single positive literal.
  - All variables assumed universally quantified
  - Examples:
    - $King(John)$
    - $King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- Comparison: Horn clauses in propositional logic!
  - Proposition symbol
  - (conjunction of symbols) $\Rightarrow$ symbol
  - Examples:
    - C
    - $A \wedge B \Rightarrow C$

Soundness of GMP

GMP is a sound inference rule! (only derives entailed sentences)

- We want to show that

$$p_1', \ldots, p_n', \ (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models \mathsf{SUBST}(\theta, q)$$

  provided that $\mathsf{SUBST}(\theta, p_i') = \mathsf{SUBST}(\theta, p_i)$ for all i
- First, for any definite clause p, we have $p \models \mathsf{SUBST}(\theta, p)$
- Second, from $p_1', \ldots, p_n'$, we can infer

$$\mathsf{SUBST}(\theta, p_1') \wedge \ldots \wedge \mathsf{SUBST}(\theta, p_n') \tag{1}$$

- Third, from $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$, we can infer

$$\mathsf{SUBST}(\theta, p_1) \wedge \ldots \wedge \mathsf{SUBST}(\theta, p_n) \Rightarrow \mathsf{SUBST}(\theta, q) \tag{2}$$

- Finally, since θ satisfies that $\mathsf{SUBST}(\theta, p_i') = \mathsf{SUBST}(\theta, p_i)$, by (1) and (2), we have $\mathsf{SUBST}(\theta, q)$

Conversion to CNF

• Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence (it is satisfiable exactly when the original sentence is satisfiable)

• Example: Everyone who loves all animals is loved by someone:

$\forall x \;\; [\forall y \;\; Animal(y) \;\Rightarrow\; Loves(x,y)] \;\Rightarrow\; [\exists y \;\; Loves(y,x)]$

1. Eliminate biconditionals and implications

$$\forall x \;\; [\neg\forall y \;\; \neg Animal(y) \vee Loves(x,y)] \vee [\exists y \;\; Loves(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x, p \equiv \exists x \;\; \neg p$, $\neg\exists x, p \equiv \forall x \;\; \neg p$:

$$\forall x \;\; [\exists y \;\; \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y \;\; Loves(y,x)]$$
$$\forall x \;\; [\exists y \;\; \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y \;\; Loves(y,x)]$$
$$\forall x \;\; [\exists y \;\; Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y \;\; Loves(y,x)]$$

Conversion to CNF contd.

3. Standardize apart variables: each quantifier should use a different variable

$$\forall x \;\; [\exists y \;\; Animal(y) \land \neg Loves(x, y)] \lor [\exists z \;\; Loves(z, x)]$$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

$$\forall x \;\; [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)]$$

Resolution: brief summary

Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\mathsf{SUBST}(\theta, (\cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots))}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.
Note $\textsc{Unify}(\alpha, \beta) = \theta$ if $\mathrm{SUBST}(\theta, \alpha) = \mathrm{SUBST}(\theta, \beta)$
For example,

- $[Animal(F(x)) \vee Loves(G(x), x)]$ and $[\neg Loves(u, v) \vee \neg Kills(u, v)]$
- We could eliminate $Loves(G(x), x)$ and $\neg Loves(u, v)$ with unifier $\theta = \{u/G(x), v/x\}$ to produce the resolvent clause $Animal(F(x)) \vee \neg Kills(G(x), x)$

Example knowledge base

- Example: build a knowledge base!

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
Prove that Colonel West is a criminal

Example knowledge base contd.

. . . it is a crime for an American to sell weapons to hostile nations:

Example knowledge base contd.

$\forall x, \exists y, z$

... it is a crime for an American to sell weapons to hostile nations:

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge$
$Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$\forall x$ $\exists y, z$

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e.,
$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$\forall x, \exists y, z$

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e.,

$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

$\forall\, x\ \ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons:

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$\forall x,$
$\exists y, z$

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\ Owns(Nono, x) \land Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

$\forall\, x\ \ Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons:

$\forall x$ $Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

$\forall x$ $\exists y, z$

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \;\Rightarrow\; Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\; Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West

$\forall\, x\;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$

Missiles are weapons:

$\forall x$ $Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

$\forall x$ $Enemy(x, America) \;\Rightarrow\; Hostile(x)$

West, who is American ...

$American(West)$

The country Nono, an enemy of America ...

$Enemy(Nono, America)$

Resolution example: West is criminal

Transfer FOL into CNF

$\forall x, \exists y, z$

- $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
  - $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
- $\forall x \;\; Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
  - $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
- $\forall x$ $Missile(x) \Rightarrow Weapon(x)$
  - $\neg Missile(x) \vee Weapon(x)$
- ...

Resolution example: West is criminal

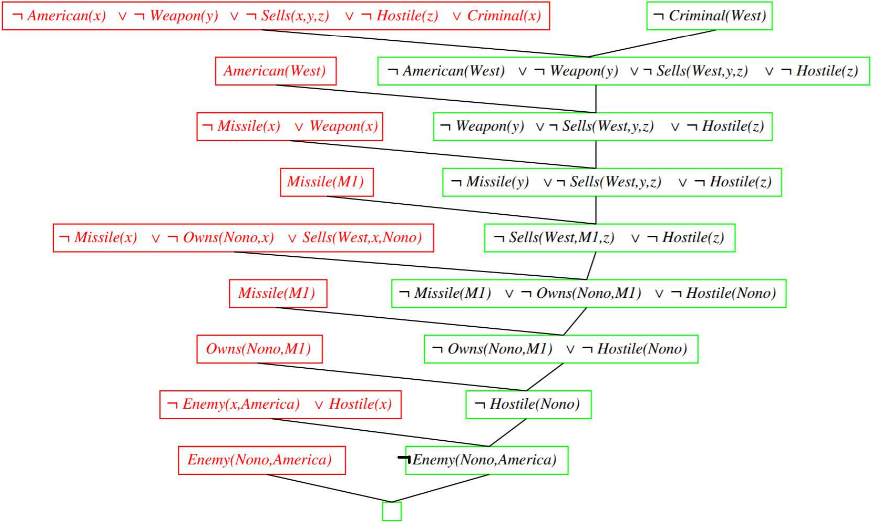
We have the following knowledge base(KB):

- $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
- $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
- $\neg Enemy(x, America) \vee Hostile(x)$
- $\neg Missile(x) \vee Weapon(x)$
- $Owns(Nono, M_1)$
- $American(West)$
- $Missile(M_1)$
- $Enemy(Nono, America)$

We want to prove $Criminal(West)$

- Apply resolution steps to $CNF(KB \wedge \neg\alpha)$
- Show $KB \wedge \neg Criminal(West)$ is unsatisfiable!

Resolution example: West is criminal



Summary

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Resolution

Acknowledgment

The slides are adapted from Stuart Russel and Marc Toussaint