TDT4136 Introduction to Artificial Intelligence

Lecture 9 - Inference in First Order Logic

Chapter 9 in the textbook

Liyuan Xing

Adjunct Associate Professor @ NAIL/IDI/NTNU
Machine Learning Engineer @ TrønderEnergi

2022

Outline

- Reduction of FOL to propositional logic (and use propositional inference methods)
- Unification and First-Order Inference
- Forward/Backward Chaining, on Definite Clauses
- Inference using Resolution, on CNF representation

Inference by Reduction to Propositionalized Sentences

- In First order logic we formulate the inference in the same way as PL
- We want to find out whether a KB entails a sentence α

Inference by Reduction to Propositionalized Sentences

- One inference approach is to propositionalize the sentences in KB and then apply inference methods (e.g., resolution refutation) used in propositional logic
  - Every FOL KB can be propositionalized so as to preserve entailment
  - A sentence is entailed by new KB iff it is entailed by the original KB

Propositionalization process:

Every KB in FOL can be "propositionalized"

- instantiate universal quantification
- instantiate existential quantification

Propositionalization

- Suppose the KB contains just the following:

$$\forall x \, King(x) \wedge Greedy(x) \implies Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

- To "propositionalize", we need to get rid of **quantifiers** and **variables**.

Propositionalization of Universal Quantification

- We instantiate the universal sentence in *all possible* ways.
  - The "Substitution" rule for instantiation of variables (for "propositionalization):

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

    variable x in the sentence α is substituted with a ground term g [1]
- Variable x is *substituted* with the *ground terms* referring to the objects *John* and *Richard* in the model one by one.

[1] A term is used to denote an object in the world. It can be a constant, variable or a Function(term1, ..., termn). A ground term is a term with no variables.

Propositionalization of Universal Quantif.- cont.

- Suppose the KB contains just the following:

  $\forall x \;\; King(x) \wedge Greedy(x) \implies Evil(x)$

  King(John)

  Greedy(John)

  Brother(Richard,John)

- How to propositionalize the first sentence?

Propositionalization of Existential Quantification

- Through **Skolemization**: Each existentially quantified variable is replaced by a *Skolem constant* or a *Skolem function*.
- *Skolem Constant*: if the existential variable is not within the scope of any universaly quantified variable. Every instance of the existentially quantified variable is replaced with the same unique constant, a brand new one that does not appear elsewhere in the knowledge base.

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

Example: $\exists y \;\; (P(y) \wedge Q(y))$ is converted to: $P(CC) \wedge Q(CC)$

Skolemization Function

- *Skolem Function*: If the existentiail quantifier is in the **scope** (i.e., "inside") of a (or more) universally quantified variable(s), then replace it with a unique n-ary function over these universally quantified variables. Remove then the existential quantifier.

E.g., $\forall x \ \exists y \ (P(x) \vee Q(y))$ converted to: $\forall x \ P(x) \vee Q(F(x))$

Problems with propositionalization

- Problem: with function symbols, there are infinitely many ground terms,
  - e.g.,Father(Father(Father(John))),etc

Solution: Herbrand Theorem

- Herbrand Theorem (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB
  - Idea: For n $= 0$ to ∞ do
    create a propositional KB by instantiating with depth-*n* terms
    see if α is entailed by this KB
- Propositionalization is complete if the sentence is entailed.
- Problem: works if α is entailed, loops if α is not entailed. Hence, it is semi-decidable
- Theorem: Turing (1936), Church (1936) Entailment for FOL is semidecidable (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

  $$\forall x \;\; King(x) \wedge Greedy(x) \implies Evil(x)$$
  $$King(John)$$
  $$Greedy(John)$$
  $$Brother(Richard, John)$$

- It seems obvious that $Evil(John)$ will be inferred at the end, 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

Inference without propositionalization

- Problem : Universal elimination gives us (too) many opportunities for substituting variables with ground terms
- Solution: avoid making blind substitution of ground terms
  - Make substitutions that help to advance inferences
  - i.e., use substitutions matching "similar" sentences in KB
- How?
- UNIFICATION: takes two similar sentences and computes the **unifier** for them (a **substitution**) that makes them look the same, if it exists UNIFY(P,Q) $= \theta$; where SUBST (θ, P)= SUBST (θ, Q)

Motivating example for Unification

- Ground clauses are clauses with no variables in them. For ground clauses we can use syntactic identity to detect when we have a P and $\neg$P pair.
- What about variables? For example, can the following two clauses be resolved?
  - P(john) $\vee$ Q(fred) $\vee$ R(x))
  - $\neg$ P(y)$\vee$ R(susan)$\vee$ R(y)

Adapting Modus Ponens to FOL

Modus Ponens in propositional logic:

$$\frac{\alpha \rightarrow \beta \quad \alpha}{\beta}$$

Generalized Modus Ponens - FOL

$$\frac{P_1', \ P_2', \ \ldots, \ P_n', \ (P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q)}{Q\theta}$$

where $P_i'\theta = P_i\theta$ for all i , θ is Substitution

Adapting Modus Ponens to FOL

- Suppose this KB:

$$S1 : \forall x \;\; King(x) \wedge Greedy(x) \implies Evil(x)$$
$$S2 : King(John)$$
$$S3 : \forall y \;\; Greedy(y)$$

- We want to find if John is Evil.
- Make a **substitution** for values of variables that make the premise of the implication identical to the sentences in the knowledge base.
  - $P_1{}'$ is King(John) and P_1 is King(x)
  - $P_2{}'$ is Greedy(y) and P_2 is Greedy(x)
  - θ is $\{x/John, y/John\}$,
  - Q is Evil(x)
  - SUBS(θ,Q) is Evil(John)

Lifted Inference Rules and Unification

- Generalized Modus ponens is the *lifted* version of Modus Ponens(i.e., variable free version)
- Lifted Inferences need finding substitutions that make 2 sentences look identical.
- This is called **Unification**

Unification process

- Unify procedure: Unify(P,Q) takes two atomic (i.e. single predicates) sentences P and Q and returns a substitution that makes P and Q identical.
- The aim is be able to match literals even when they have variables.
- Rules for substitutions: Can replace a variable
  - by a constant.
  - by a variable.
  - by a function expression, as long as the function expression does not contain the variable.

Unifier: a substitution that makes two clauses resolvable. e.g., θ:$\{x_1/M; x_2/x_3; x_4/F(..)\}$

More on Unification

- Not all formulas can be unified - substitutions only affect variables.
  - Example: Consider the pair P(F(x),A) and P(y,F(w))
  - This pair cannot be unified as there is no way of making A = F(w) with a substitution.
- In θ, each variable is paired at most once
- A variable's pairing term may not contain the variable directly or indirectly.
  - e.g. can't have substitution { x/F(y), y/F(x) }
- When unifying expressions P and Q, the variable names in P and the variable names in Q should be disjoint.
  - Yes:UNIFY(Loves (John, x), Loves (y, Jane)) θ={ x/Jane, y/John }
  - No:UNIFY(Loves (John, x), Loves (x, Jane)) – No unifier

Unification-exercise

P	Q	Θ
Knows(John, x)	Knows(John,Jane)	
Knows(John, x)	Knows(y, OJ)	
Knows(John, x)	Knows(y, Mother(y))	

Unification-exercise

P	Q	Θ
Knows(John, x)	Knows(John,Jane)	{x/Jane}}
Knows(John, x)	Knows(y, OJ)	{x/OJ, y/John}
Knows(John, x)	Knows(y, Mother(y))	{x/Mother(John), y/John}

Unify and Infer

P	Q	Θ
Knows(John, x)	Knows(John,Jane)	{x/Jane}}
Knows(John, x)	Knows(y, OJ)	{x/OJ, y/John}
Knows(John, x)	Knows(y, Mother(y))	{x/Mother(John), y/John}

Idea: Unify rule premises with known facts, apply unifier to conclusion
E.g., if we know both Q and Knows(John,x) $\rightarrow$ Likes(John,x)
then we conclude

Likes(John,Jane)
Likes(John,OJ)
Likes(John,Mother(John))

Most General Unifier

- Our aim is to be able to match conflicting literals (for the use of resolution), even when they have variables. Unification process determines whether there is a"specialization" that matches.
- However, we don't want to over specialize.

Most General Unifier-example

- Consider the two sentences:

  $\neg P(x) \vee S(x) \vee Q(April)$
  P(y)$\vee$ R(y)

- Possible unifications:

  (S(Arvid)$\vee$ Q(April)$\vee$ R(Arvid)) {y=x, x=Arvid}
  (S(Sophie) $\vee$ Q(April)$\vee$ R(Sophie)) {y=x, x=Sophie}
  (S(x)$\vee$ Q(April)$\vee$ R(x)) {y=x}

- The last unifier is the "most-general" one, the other two are specializations of it.
- We want to keep the most general clause so that we can use it in future resolution steps.

Most General Unifier

- Informally, the most general unifier (MGU) imposes the fewest constraints on the terms (contains the most variables).
- Formally, a substitution θ is more general than β iff there is a substitution σ such that $\theta\ \sigma = \beta$.
  e.g. θ = z/F(w) is more general than β = z/F(C) since σ = w/C

Motivation for Standardizing apart

P	Q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ, y/John}
Knows(John,x)	Knows(y,Mother(y))	{x/Mother(John),y/John}
Knows(John,x)	Knows(x,OJ)	?

Motivation for Standardizing apart

P	Q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ, y/John}
Knows(John,x)	Knows(y,Mother(y))	{x/Mother(John),y/John}
Knows(John,x)	Knows(x,OJ)	No substitution possible yet (i.e., "fails")

Standardizing apart

Knows (John, x) and Knows (x, OJ) cannot be unified, i.e, unifications *fails*. Needs "**Standardizing apart**".

- Intuitively we know that if we know: John hates everyone he knows and that everyone knows OJ. So we should be able to infer that John hates OJ.
- This is why we require that every variable has a separate name.
- UNIFY $Knows(John, z_{27})$ and $Knows(z_{17}, OJ)$ This works!!! $\{z_{17}/John, z_{27}/OJ\}$
- **Standardizing apart** eliminates overlap of variables.

Inference through Forward Chaining in FOL

- Forward Chaining is an important inference in FOL - without propositionalisation
- FC uses definite clauses. In FOL
  - A definite clause: either atomic, or implication
  - Existential quantifiers are not allowed
  - Universal quantifications are implicit.
  - Example sentences in DC:

    King (John) - literal
    King(x) $\rightarrow$ Evil(x)
    Evil(x). (i.e., everyone is evil). - literal with variable

Example on Forward Chaining in FOL

Suppose we have the following knowledge in the KB:

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.

We want to Prove that **"Colonel West is a criminal".**

Colonel West example cont.

- KB: 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.
- We need
  - to translate these natural language sentences to FOL
  - convert these sentences in the KB to Definite Clauses if they are not in DC form by eliminating quantifiers:
    - remove $\forall$
    - eliminate $\exists$ by skolemization

Example knowledge base contd.

S1: It is a crime for an American to sell weapons to hostile nations:

Example knowledge base contd.

S1: It is a crime for an American to sell weapons to hostile nations:

$\forall x, y, z \; \mathit{American}(x) \wedge \mathit{Weapon}(y) \wedge \mathit{Sells}(x, y, z) \wedge \mathit{Hostile}(z) \implies \mathit{Criminal}(x)$

S1: $\mathit{American}(x) \wedge \mathit{Weapon}(y) \wedge \mathit{Sells}(x, y, z) \wedge \mathit{Hostile}(z) \implies \mathit{Criminal}(x)$

S2: Nono . . . has some missiles

Example knowledge base contd.

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

S1: $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

S2: Nono ... has some missiles

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

S2: $Owns(Nono, M_1)$ and $Missile(M_1)$

S3: ... all of its missiles were sold to it by Colonel West

Example knowledge base contd.

S1: $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

S2: $Owns(Nono, M_1)$ and $Missile(M_1)$

S3: . . . all of its missiles were sold to it by Colonel West

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

S3: $Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$

S4: Missiles are weapons:

Example knowledge base contd.

S1: $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

S2: $Owns(Nono, M_1)$ and $Missile(M_1)$

S3: $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$

S4: Missiles are weapons.

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

S4: $Missile(x) \Rightarrow Weapon(x)$

S5: An enemy of America counts as "hostile":

Example knowledge base contd.

S1: $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

S2: $Owns(Nono, M_1)$ and $Missile(M_1)$

S3: $Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$

S4: $Missile(x) \Rightarrow Weapon(x)$

S5: An enemy of America counts as "hostile"

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

S5: $Enemy(x, America) \implies Hostile(x)$

S6: West, who is American ...

S6: $American(West)$

S7: The country Nono, an enemy of America ...

S7: $Enemy(Nono, America)$

Forward chaining algorithm - abstract

- Similar to Propositional Logic, restriction on sentences - definite clause.
- Find and use rules that have as premises the know facts in the KB.
  - In Propositional Logic: $A \wedge B \implies C$
  - In FOL: $King(x) \wedge Greedy(x) \implies Evil(x)$
  - FOL Needs Unification to match sentences.
- Continue matching and deriving consequences of sentences until deriving the goal.

Forward chaining proof of Colonel West

Our KB:

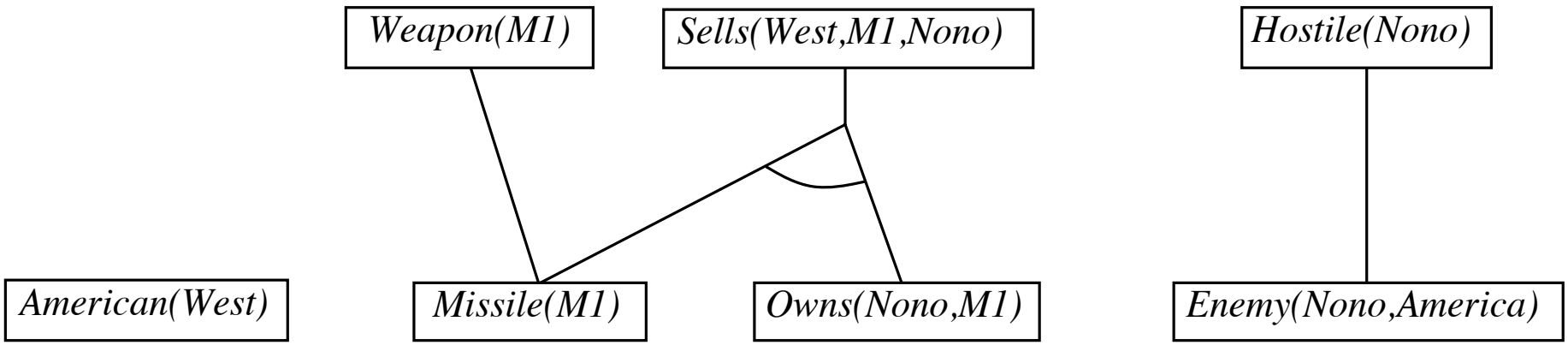
1. $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
2. $Owns(Nono, M_1)$
3. $Missile(M_1)$
4. $Missile(x) \Rightarrow Weapon(x)$
5. $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
6. $Enemy(x, America) \implies Hostile(x)$
7. $American(West)$
8. $Enemy(Nono, America)$

Forward chaining proof of Colonel West

American(West) | *Missile(M1)* | *Owns(Nono,M1)* | *Enemy(Nono,America)*

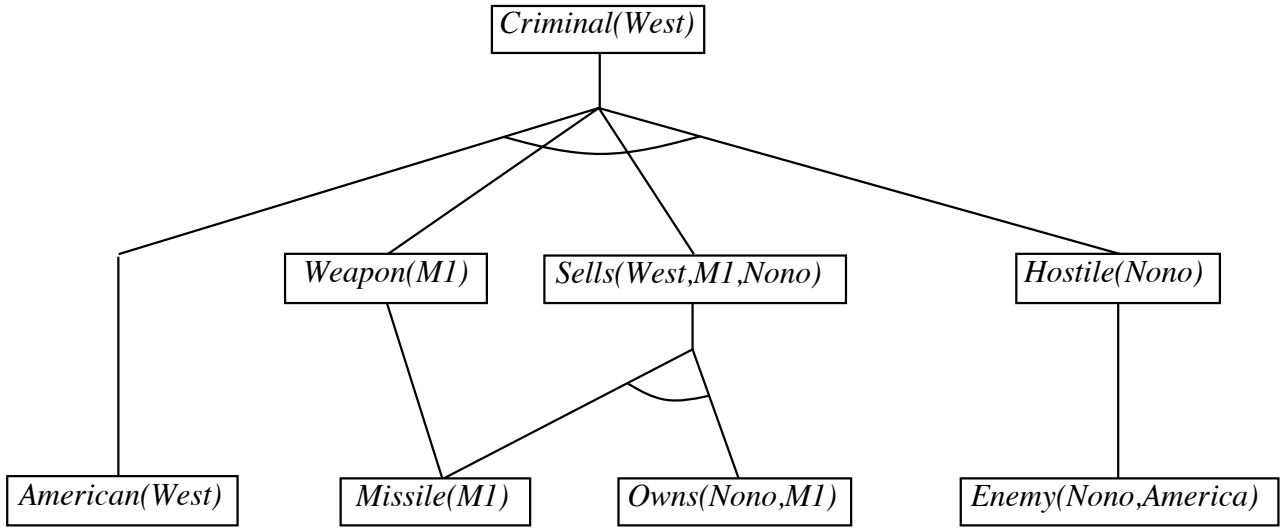
1. $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
2. $Owns(Nono, M_1)$
3. $Missile(M_1)$
4. $Missile(x) \Rightarrow Weapon(x)$
5. $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
6. $Enemy(x, America) \implies Hostile(x)$
7. $American(West)$
8. $Enemy(Nono, America)$

Forward chaining proof



1. $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
2. $Owns(Nono, M_1)$
3. $Missile(M_1)$
4. $Missile(x) \Rightarrow Weapon(x)$
5. $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
6. $Enemy(x, America) \implies Hostile(x)$
7. $American(West)$
8. $Enemy(Nono, America)$

Forward chaining proof



1. $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
2. $Owns(Nono, M_1)$
3. $Missile(M_1)$
4. $Missile(x) \Rightarrow Weapon(x)$
5. $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
6. $Enemy(x, America) \implies Hostile(x)$
7. $American(West)$
8. $Enemy(Nono, America)$

Forward Chaining and efficiency

- checks every rule against every fact

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

  e.g., order the conjuncts to be checked in the premise. Heuristic: Check the one with fewest sentence in the KB first -reminds the *Minimum remaining values* heuristic in CSP.

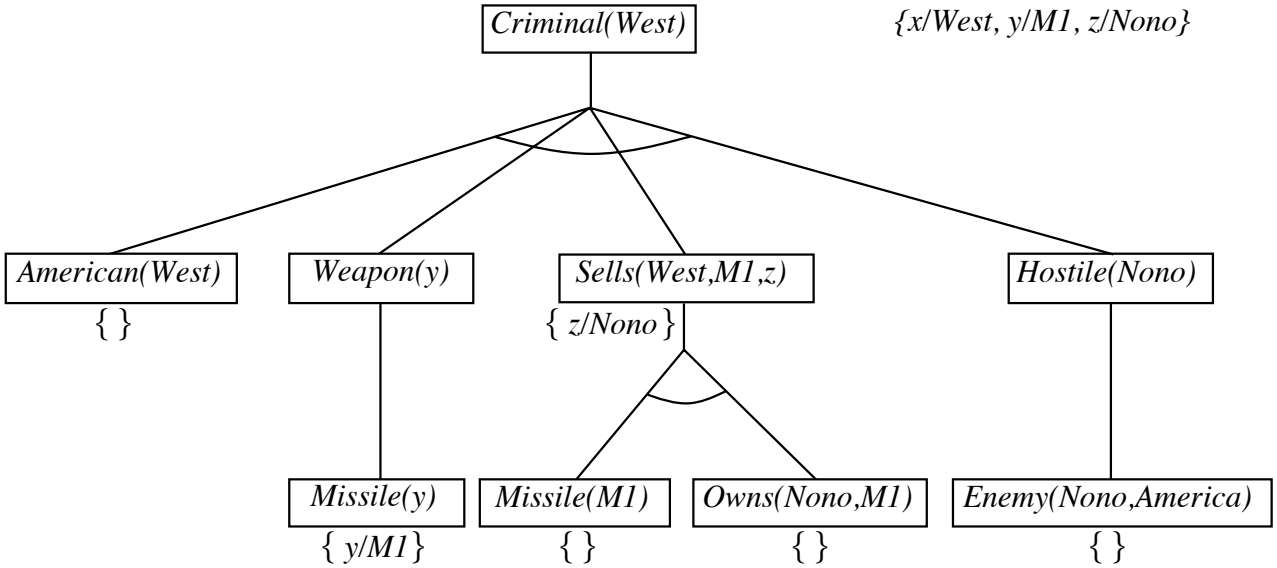
- rechecks every rule in every iteration

  e.g., eliminate redundant rule-matching through incremental FC: Check a rule at iteration t if only its premise includes a conjunct P1 that unifies with a fact P1′ inferred at iteration t-1

- can generate many facts irrelevant to the goal

  e.g., do Backward Chaining instead

Backward chaining example



1. $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
2. $Owns(Nono, M_1)$
3. $Missile(M_1)$
4. $Missile(x) \Rightarrow Weapon(x)$
5. $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
6. $Enemy(x, America) \implies Hostile(x)$
7. $American(West)$
8. $Enemy(Nono, America)$

Example Resolution Refutation - same example

Now we solve the same Colonel West example using resolution, more correctly resolution refutation.

First, we look at Resolution rule in FOL.

Resolution Rule

Full first-order version:

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

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$.

Two standardized clauses can be resolved if they contain complementary literals (one is the negation of the other).

FOL literals are complementary if one *unifies* with the negation of the other.

Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

Conversion to CNF - example

We'll soon start solving Colonel West example using Resolution Refutation.

- The idea is the same as in Propositional Logic:
  Our goal is to determine if KB $\models \alpha$:
  1. Add $\neg\alpha$ to the KB
  2. Convert KB and α to Conjunctive Normal Form
  3. Use the resolution rule and search to determine whether the system is satisfiable (SAT)
- Let us look at the procedure for conversion of FOL sentences to CNF through an example.

Conversion to CNF - example

Translate this sentence to FOL and convert to CNF:

"Everyone who loves all animals is loved by someone"

Conversion to CNF - example

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

Conversion to CNF - example

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

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

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

Conversion to CNF contd.

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

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

4. Skolemize: Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$$\forall x \ [Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$$

6. Distribute $\wedge$ over $\vee$:

$$[Animal(f(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$$

Colonel West example using Resolution Refutation

- KB:
  $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$
  $Owns(Nono, M_1)$ and $Missile(M_1)$
  $Missile(x) \Rightarrow Weapon(x)$
  $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
  $Enemy(x, America) \implies Hostile(x)$
  $American(West)$
  $Enemy(Nono, America)$

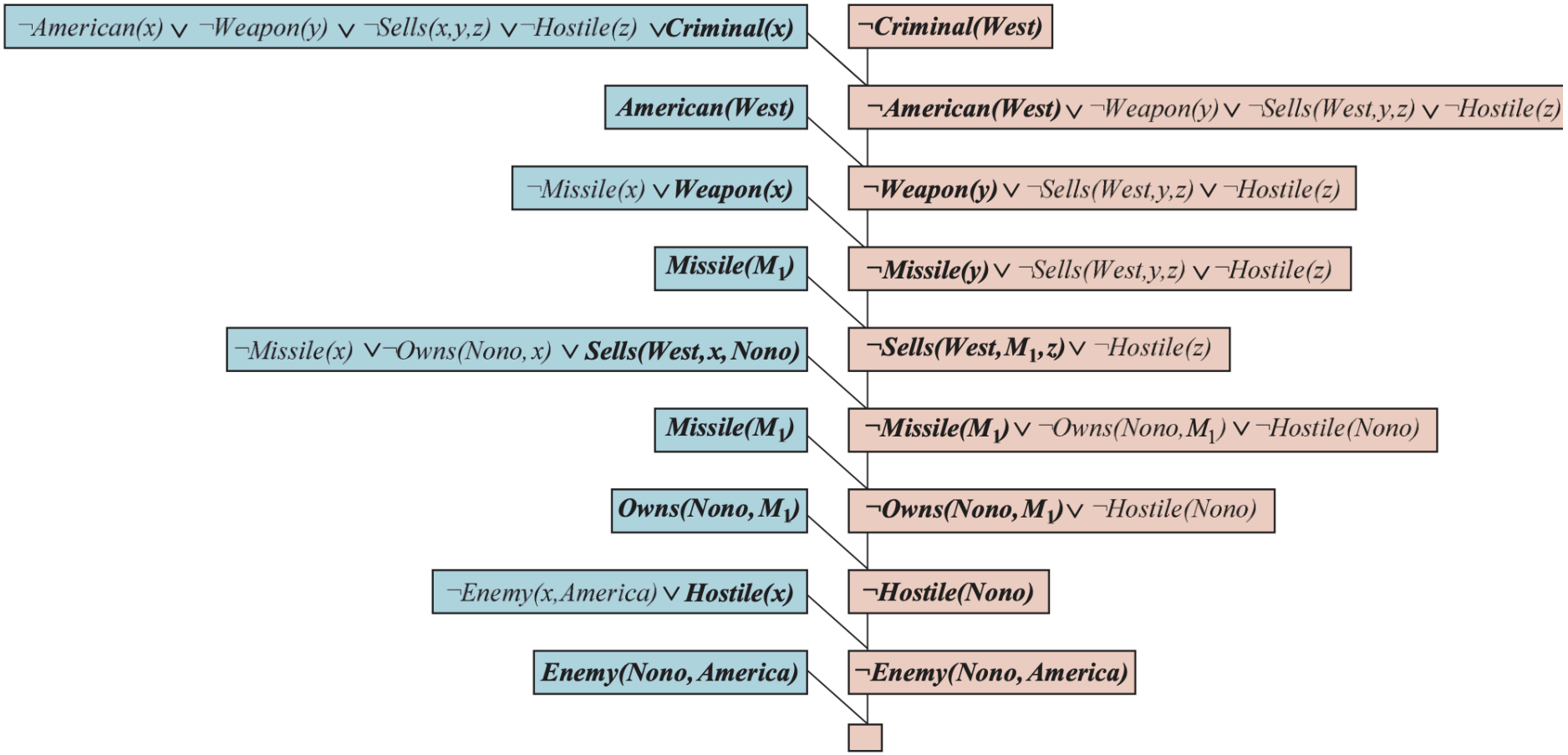
- Question: Is colonel West criminal?

Colonel West example in CNF form

1. $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
2. $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
3. $\neg Enemy(x, America) \vee Hostile(x)$
4. $\neg Missile(x) \vee Weapon(x)$

- Facts:
  $American(West), Owns(Nono, M1), Missile(M1), Enemy(Nono, America)$
- Add $\neg Criminal(West)$

Colonel West example using Resolution Refutation - cont



Efficient Resolution Strategies

- Unit preference: prefer inferences where one of the sentences is a single literal - towards shorter clauses, finaly empty one
- Set of support: start with the negated query, and add resolvents into the set. In each inference use one of the sentences from this set - search space is reduced.
- Input resolution: In each inference use one of the sentences in the original KB or query, and one other sentence (i.e., a resolvent). Complete for Horn clauses.

More on Skolemization -Examples

- $\exists$ x Sibling (sofie, x)
  Skolemized: Sibling(Sofie,SkSister)
- $\forall$ x $\exists y$ Parent(x,y)
  Skolemized: $\forall$ x Parent(x, F(x))
- $\forall$ x,y Grandpa(x,y) $\implies$ $\exists$ z Parent (x,z) $\wedge$ Parent(z,y)
  Skolemized: $\forall$ x,y Grandpa(x,y) $\implies$ Parent(x, F(x,y)) $\wedge$ Parent(F(x,y), y)
- $\forall$ x [$\exists y$ Animal(y) $\wedge\neg$ Loves(x,y)] $\vee$ [$\exists$ z Loves(z,x)]
  Skolemized: $\forall$ x [Animal(F(x)) $\wedge$ $\neg$ Loves(x, F(x))] $\vee$ Loves (G(x), x)