Lecture 29

CS 6380 Artificial Intelligence

Inference in First-Order Logic (contd..)

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India



Roadmap: Four Families of First-Order Inference

- Propositionalization: reduce FOL to propositional logic and reuse Chapter 7
- Unification and lifting: inference rules that work on FOL sentences directly
- Forward chaining: data-driven; deductive databases, production systems
- Backward chaining: goal-driven; logic programming and Prolog
- Resolution: refutation-complete for all of FOL; theorem provers

Forward and backward chaining handle *definite (Horn) clauses* only. Resolution handles any FOL sentence

First-Order Definite Clauses

- Disjunction of literals with exactly one positive literal
- Either an atom, or conjunction of positive literals ⇒ one positive literal
- Variables are implicitly universally quantified
- Datalog: definite clauses with *no function symbols* (like a relational database)

Definite clauses

$\text{Attends}(x, \text{IITM}) \wedge \text{Studies}(x, \text{AI}) \Rightarrow \text{GoodAt}(x, \text{AI})$

$\text{Attends}(\text{Alfred}, \text{IITM})$

$\text{Studies}(y, \text{AI})$

Not definite

$\text{Attends}(x, \text{IITM}) \Rightarrow \text{GoodAt}(x, \text{AI}) \vee \text{GoodAt}(x, \text{Bio})$

two positive literals in its clause form

Running Example: The Crime Problem

"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."

English	Definite clause
(9.3) Crime to sell weapons to hostile nations	$\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$
(9.4, 9.5) Nono has some missiles (EI)	$\text{Owns}(\text{Nono}, M_1), \quad \text{Missile}(M_1)$
(9.6) All its missiles were sold by West	$\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
(9.7) Missiles are weapons	$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
(9.8) An enemy of America is hostile	$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$
(9.9) West is American	$\text{American}(\text{West})$
(9.10) Nono is an enemy of America	$\text{Enemy}(\text{Nono}, \text{America})$

Goal: $\text{Criminal}(\text{West})$. No function symbols, so this is a Datalog KB

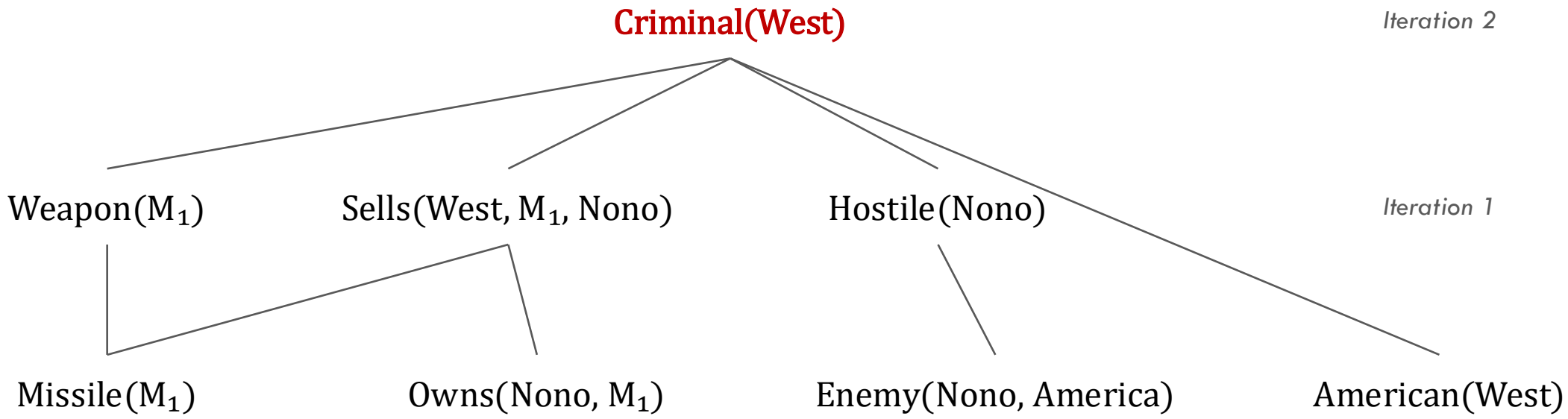
A Simple Forward-Chaining Algorithm

```
function FOL-FC-ASK(KB, α) returns a substitution or false
  repeat until new is empty
    new ← { }
    for each rule in KB do
      (p1 ∧ … ∧ pn ⇒ q) ← STANDARDIZE-VARIABLES(rule)
      for each θ such that SUBST(θ, p1 ∧ … ∧ pn)
                          = SUBST(θ, p1' ∧ … ∧ pn')
                  for some p1', …, pn' in KB
        q' ← SUBST(θ, q)
        if q' does not unify with a sentence
              already in KB or new then
          add q' to new
          φ ← UNIFY(q', α)
          if φ is not fail then return φ
    add new to KB
  return false
```

- Fire every rule whose premises hold
- Add the conclusions
- Stop at the answer, or at a fixed point
- A renaming is not new:

  Likes(x, IceCream)
  Likes(y, IceCream)

Forward Chaining on the Crime Example



Iteration 1

(9.6) $\{x/M_1\} \rightarrow$ Sells(West, M_1, Nono)
(9.7) $\{x/M_1\} \rightarrow$ Weapon(M_1)
(9.8) $\{x/\text{Nono}\} \rightarrow$ Hostile(Nono)

Iteration 2

(9.3) $\{x/\text{West}, y/M_1, z/\text{Nono}\} \rightarrow$ Criminal(West)
Nothing new can follow: a fixed point

Properties of Forward Chaining

Sound

Every step is Generalized Modus Ponens

Complete

For any knowledge base of definite clauses

Terminates on Datalog

At most $p \cdot n^k$ ground facts can ever be added

p predicates, n constants, maximum arity k

With function symbols it may never stop

- Peano axioms: $\text{NatNum}(0), \quad \forall n\ \text{NatNum}(n) \Rightarrow \text{NatNum}(S(n))$
- Forward chaining adds $\text{NatNum}(S(0))$, $\text{NatNum}(S(S(0)))$, … forever
- Unavoidable: entailment with definite clauses is semidecidable

Efficiency (1): Matching Rules Against Facts

- Pattern matching in the inner loop can be expensive

  $\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

  - Nono owns many things but few missiles? Find missiles first
- Conjunct ordering: optimal order is NP-hard; the MRV heuristic helps
- Any finite-domain CSP is one definite clause, so matching is NP-hard

Map colouring as a clause

$\text{Diff}(wa, nt) \land \text{Diff}(wa, sa) \land$
$\text{Diff}(nt, q) \land \text{Diff}(nt, sa) \land$
$\text{Diff}(q, nsw) \land \text{Diff}(q, sa) \land$
$\text{Diff}(nsw, v) \land \text{Diff}(nsw, sa) \land$
$\text{Diff}(v, sa) \Rightarrow \text{Colorable}()$

Facts: Diff(Red, Blue), Diff(Red, Green), …

Why Matching Is Usually Fine

Small rules

Rule size and arity are bounded in practice, so data complexity is polynomial in the number of facts

Tractable subclasses

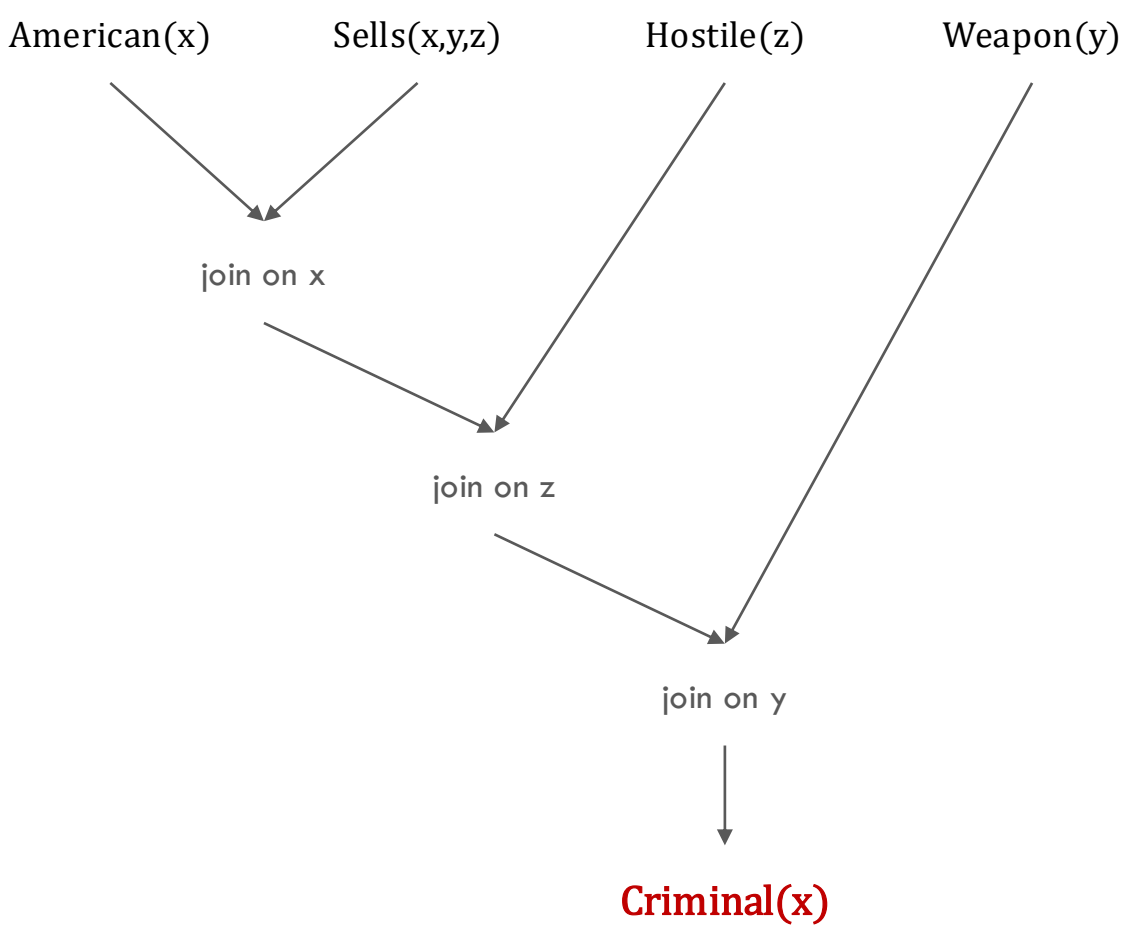
A tree-shaped constraint graph makes matching linear time, as for tree-structured CSPs

Less repeated work

Incremental forward chaining avoids re-matching the same rules every iteration

Efficiency (2): Incremental Chaining and Rete

- Every new fact needs a fact that was new one step earlier
  - So only check rules that touch a newly inferred fact
- Rete ("net"): a dataflow network that keeps partial matches
- Production systems: XCON / R1 configured DEC computers with thousands of rules
- Cognitive architectures: ACT, SOAR



Efficiency (3): Irrelevant Facts

- Forward chaining derives *everything*, relevant to the goal or not
- Fixes: backward chaining (next), restricting the rule set, or magic sets
- Magic sets (deductive databases): rewrite rules using the goal $\text{Criminal}(\text{West})$

  $\text{Magic}(x) \wedge \text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

  - Add $\text{Magic}(\text{West})$: among millions of Americans, only West is considered
- A hybrid: *backward* preprocessing, then *forward* inference

A Backward-Chaining Algorithm

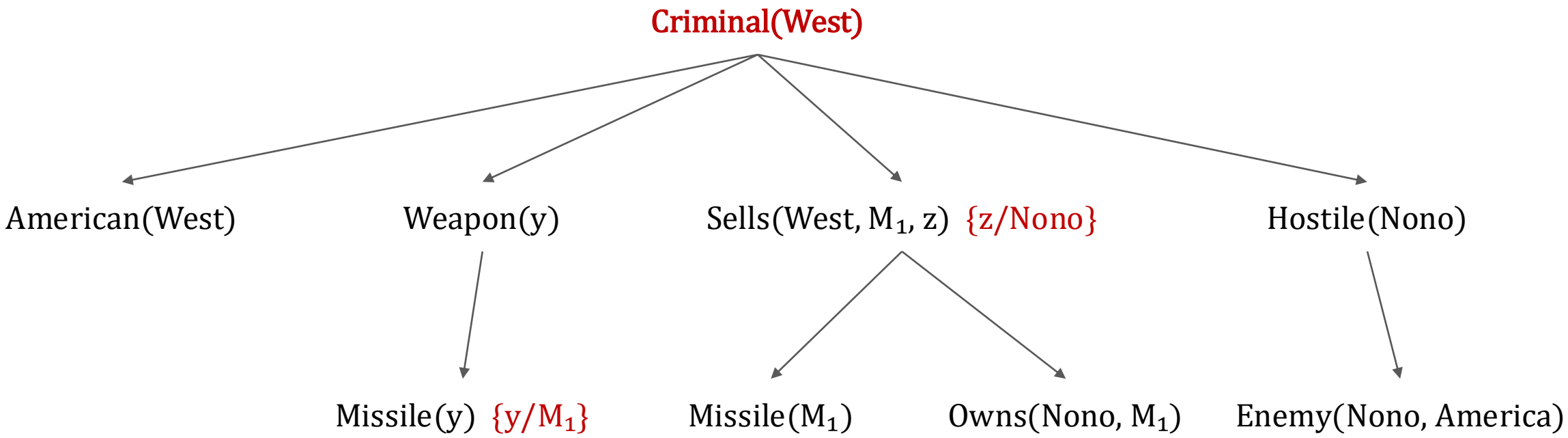
```
function FOL-BC-ASK(KB, query) returns a generator
  return FOL-BC-OR(KB, query, { })

generator FOL-BC-OR(KB, goal, θ)
  for each rule (lhs ⇒ rhs) in
            FETCH-RULES-FOR-GOAL(KB, goal) do
    (lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ' in FOL-BC-AND(KB, lhs,
                      UNIFY(rhs, goal, θ)) do
      yield θ'

generator FOL-BC-AND(KB, goals, θ)
  if θ = failure then return
  else if LENGTH(goals) = 0 then yield θ
  else
    first, rest ← FIRST(goals), REST(goals)
    for each θ' in FOL-BC-OR(KB, SUBST(θ, first), θ)
      for each θ" in FOL-BC-AND(KB, rest, θ')
        yield θ"
```

- AND/OR search
  - OR: any rule can prove the goal
  - AND: every conjunct must be proved
- A generator: yields one substitution at a time
- Depth-first: linear space, but repeated states and incompleteness

Backward Chaining on the Crime Example



- Read depth first, left to right. Each binding carries forward: when we reach the last conjunct, originally Hostile(z), z is already Nono

Conjunctive Normal Form for First-Order Logic

- Resolution needs CNF: a conjunction of clauses, each a disjunction of literals
- Variables in clauses are universally quantified
- Every FOL sentence has an inferentially equivalent CNF form: unsatisfiable exactly when the original is
- New compared with Chapter 7: eliminating existential quantifiers
- Example: the crime rule

$$\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$$
$$\neg\text{American}(x) \lor \neg\text{Weapon}(y) \lor \neg\text{Sells}(x, y, z) \lor \neg\text{Hostile}(z) \lor \text{Criminal}(x)$$

Converting to CNF: A Worked Example

"Everyone who loves all animals is loved by someone."

$$\forall x\,[\forall y\,\text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y\,\text{Loves}(y, x)]$$

Step	Result
1. Eliminate ⇒	$\forall x\,[\neg\forall y\,\neg\text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y\,\text{Loves}(y, x)]$
2. Move ¬ inwards	$\forall x\,[\exists y\,\text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists y\,\text{Loves}(y, x)]$
3. Standardize variables	$\forall x\,[\exists y\,\text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists z\,\text{Loves}(z, x)]$
4. Skolemize	$\forall x\,[\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$
5. Drop ∀	$[\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$
6. Distribute ∨ over ∧	$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg\text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

In step 2: $\neg\forall x\,p \equiv \exists x\,\neg p$, $\neg\exists x\,p \equiv \forall x\,\neg p$

Skolemization: Getting It Right

Wrong: blind Existential Instantiation

$\forall x\ [\text{Animal}(A) \wedge \neg\text{Loves}(x, A)] \vee \text{Loves}(B, x)$

Everyone fails to love the *same* animal A, or is loved by the *same* B

Right: Skolem functions

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

$F(x)$: an animal x might not love

$G(x)$: someone who might love x

- Skolem function arguments = all universally quantified variables whose scope contains the ∃
- Like EI, the result is satisfiable exactly when the original is
- Back to our vocabulary: $\forall x\ \exists y\ \text{Attends}(x, y)$ becomes $\forall x\ \text{Attends}(x, U(x))$

The Resolution Inference Rule

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

where UNIFY(ℓ_i, $\neg m_j$) = θ

Example

$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)]$

$[\neg\text{Loves}(u, v) \vee \neg\text{Kills}(u, v)]$

$\theta = \{u/G(x), v/x\}$

$\Rightarrow [\text{Animal}(F(x)) \vee \neg\text{Kills}(G(x), x)]$

- Standardize the clauses apart first
- Literals are complementary if one unifies with the other's negation
- Binary resolution alone is incomplete; add factoring

Resolution Proof 1: West Is a Criminal

Resolve the spine with	Resolvent
Negated goal	¬Criminal(West)
¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x,y,z) ∨ ¬Hostile(z) ∨ Criminal(x)	¬American(West) ∨ ¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
American(West)	¬Weapon(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ Weapon(x)	¬Missile(y) ∨ ¬Sells(West,y,z) ∨ ¬Hostile(z)
Missile(M_1)	¬Sells(West,M_1,z) ∨ ¬Hostile(z)
¬Missile(x) ∨ ¬Owns(Nono,x) ∨ Sells(West,x,Nono)	¬Missile(M_1) ∨ ¬Owns(Nono,M_1) ∨ ¬Hostile(Nono)
Missile(M_1)	¬Owns(Nono,M_1) ∨ ¬Hostile(Nono)
Owns(Nono,M_1)	¬Hostile(Nono)
¬Enemy(x,America) ∨ Hostile(x)	¬Enemy(Nono,America)
Enemy(Nono,America)	□ empty clause

The spine is exactly the goal sequence of backward chaining, a **special case of resolution**

Resolution Proof 2: Did Curiosity Kill the Cat?

- Everyone who loves all animals is loved by someone.
- Anyone who kills an animal is loved by no one.
- Jack loves all animals.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- **Did Curiosity kill the cat?**

	Clauses in CNF
A1	$\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)$
A2	$\neg\text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)$
B	$\neg\text{Loves}(y, x) \vee \neg\text{Animal}(z) \vee \neg\text{Kills}(x, z)$
C	$\neg\text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)$
D	$\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$
E	$\text{Cat}(\text{Tuna})$
F	$\neg\text{Cat}(x) \vee \text{Animal}(x)$ (cats are animals)
¬G	$\neg\text{Kills}(\text{Curiosity}, \text{Tuna})$

Curiosity Proof: The Resolution Steps

Resolve	Resolvent
E with F	Animal(Tuna)
D with ¬G	Kills(Jack, Tuna)
Animal(Tuna) with B	¬Loves(y, x) ∨ ¬Kills(x, Tuna)
… with Kills(Jack, Tuna)	¬Loves(y, Jack)
A2 with C (standardized apart)	¬Animal(F(Jack)) ∨ Loves(G(Jack), Jack)
… with A1, then factor	Loves(G(Jack), Jack)
¬Loves(y, Jack) with that	□ contradiction

In English

If Curiosity didn't, Jack killed Tuna.

Tuna is a cat, so an animal.

So no one loves Jack.

But Jack loves all animals, so someone loves him.

Contradiction.

Extracting Answers: Who Killed the Cat?

- Goal $\exists w\ \mathrm{Kills}(w, \mathrm{Tuna})$; negated: $\neg\mathrm{Kills}(w, \mathrm{Tuna})$
  - Track bindings: the proof yields $\{w/\mathrm{Curiosity}\}$
- Danger: nonconstructive proofs
  - Resolve $\neg\mathrm{Kills}(w, \mathrm{Tuna})$ with D, then again: w is bound two ways
  - "Someone killed Tuna: Jack or Curiosity." Not news!
- Fix 1: bind each query variable once per proof, and backtrack
- Fix 2: an answer literal

$$\neg\mathrm{Kills}(w, \mathrm{Tuna}) \vee \mathrm{Answer}(w) \rightsquigarrow \mathrm{Answer}(\mathrm{Curiosity})$$

Completeness of Resolution

- Refutation-complete: if S is unsatisfiable, resolution derives a contradiction in finitely many steps
- It can't list *all* consequences, but proves $\mathrm{KB} \models \alpha$ by refuting $\mathrm{KB} \wedge \neg\alpha$
- Proof (Robinson, 1965) in three steps:

S is unsatisfiable, in clausal form

↓ *Herbrand's theorem*

Some finite set S′ of ground instances is unsatisfiable

↓ *Ground resolution theorem*

Propositional resolution finds a contradiction in S′

↓ *Lifting lemma*

A first-order resolution proof exists for S

Handling Equality

Axiomatize it

- Reflexive, symmetric, transitive, plus substitution axioms
- Many useless conclusions

Demodulation

- Rewrite with $x = y$, always $x \rightarrow y$
- Paramodulation generalizes it and is complete

Equational unification

- Unify $1 + 2$ with $2 + 1$ using commutativity
- Close to constraint logic programming

Demodulation example

$$\text{Father}(\text{Father}(x)) = \text{PaternalGrandfather}(x), \quad \text{Birthdate}(\text{Father}(\text{Father}(\text{Bella})), 1926)$$

$$\Rightarrow \text{Birthdate}(\text{PaternalGrandfather}(\text{Bella}), 1926)$$

Resolution Strategies

- Unit preference: resolve with single-literal clauses first; resolvents get shorter
  - Complete for Horn clauses; OTTER generalizes it with clause weights
- Set of support: every step uses a clause from a chosen set, e.g. the *negated query*
  - Goal-directed; complete if the rest of the KB is satisfiable
- Input resolution: each step uses an input clause; complete for Horn only
  - Linear resolution also allows ancestors and is complete
- Subsumption: with $P(x)$ known, drop $P(A)$ and $P(A) \vee Q(B)$

Theorem Provers in Practice

- Robbins algebra (open since 1933): EQP proved it in 1996 after **8 days** of search
- Hardware: whole CPUs verified, including timing; FM9001 proved correct with NQTHM
- Software: RSA and Boyer–Moore string matching verified; spacecraft code checked with SPIN
- Synthesis: prove “there exists a program p meeting this spec” (Green, 1969)
- Mathematics: first fully formal proof of Gödel's theorem (Shankar, 1986)

Summary

- Propositionalization: complete, but slow unless the domain is small
- Unification finds just the substitutions a proof needs
- Generalized Modus Ponens: complete for definite clauses; decidable for Datalog
- Forward chaining: deductive databases, production systems
- Backward chaining: logic programming; loops and redundancy, fixed by tabling
- Resolution on CNF: complete for FOL; strategies and paramodulation make it practical