# Lecture 15 of 42

## FOL: Resolution Strategies and Computability
## Discussion: Decidability and Logic

**William H. Hsu**
**Department of Computing and Information Sciences, KSU**

**KSOL course page:** **http://snipurl.com/v9v3**
**Course web site:** **http://www.kddresearch.org/Courses/CIS730**
**Instructor home page:** **http://www.cis.ksu.edu/~bhsu**

**Reading for Next Class:**
Chapter 9, p. 272 - 319, Russell & Norvig 2nd edition
Definition of Type-0 Grammar: **http://en.wikipedia.org/wiki/Chomsky_hierarchy**
Russell's Paradox: **http://en.wikipedia.org/wiki/Russell%27s_paradox**
Logic programming: **http://en.wikipedia.org/wiki/Logic_programming**



# Lecture Outline

- **Reading for Next Class: Review Chapter 9 (p. 272 – 319), R&N 2e**
- **Last Class: Forward/Backward Chaining, 9.2-9.4 (p. 275-294), R&N 2e**
  - **Unification (previewed in Lecture 13)**
  - **GMP implemented: forward chaining / Rete algorithm, backward chaining**
  - **Conversion to clausal form (CNF): procedure, example**
  - **Prolog: SLD resolution**
  - **Preview: refutation theorem proving using resolution, proof example**
- **Today: Resolution Theorem Proving, Section 9.5 (p. 275-294), R&N 2e**
  - **Proof example in detail**
  - **Paramodulation and demodulation**
  - **Resolution strategies: unit, linear, input, set of support**
  - **FOL and computability: complements and duals**
    - ⇨ **Semidecidability ($\in$ RE \ REC) of validity, unsatisfiability**
    - ⇨ **Undecidability of ($\notin$ RE) of non-validity, satisfiability**
  - **Theoretical foundations and ramifications of decidability results**
- **Next Class: Logic Programming (Prolog), Knowledge Engineering**

# FOL Sequent Rules and Example: Review

- | | |
  |---|---|
  | Bob is a buffalo | 1. $Buffalo(Bob)$ |
  | Pat is a pig | 2. $Pig(Pat)$ |
  | Buffaloes outrun pigs | 3. $\forall x, y \;\; Buffalo(x) \land Pig(y) \Rightarrow Faster(x, y)$ |
  | Bob outruns Pat | |

- Apply Sequent Rules to Generate New Assertions

| | |
|---|---|
| AI 1 & 2 | 4. $Buffalo(Bob) \land Pig(Pat)$ |
| UE 3, $\{x/Bob, y/Pat\}$ | 5. $Buffalo(Bob) \land Pig(Pat) \Rightarrow Faster(Bob, Pat)$ |
| MP 6 & 7 | 6. $Faster(Bob, Pat)$ |

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta} \qquad \frac{\alpha \quad \beta}{\alpha \land \beta} \qquad \frac{\forall x \;\; \alpha}{\alpha\{x/\tau\}}$$

- Modus Ponens — And Introduction — Universal Elimination





# Forward Chaining in FOL – Algorithm: Review

```
function FOL-FC-ASK(KB, α) returns a substitution or false
   repeat until new is empty
       new ← { }
       for each sentence r in KB do
           (p1 ∧ ... ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
           for each θ such that (p1 ∧ ... ∧ pn)θ = (p'1 ∧ ... ∧ p'n)θ
                        for some p'1, ..., p'n in KB
               q' ← SUBST(θ, q)
               if q' is not a renaming of a sentence already in KB or new then do
                     add q' to new
                     φ ← UNIFY(q', α)
                     if φ is not fail then return φ
       add new to KB
   return false
```

# Backward Chaining in FOL – Algorithm: Review

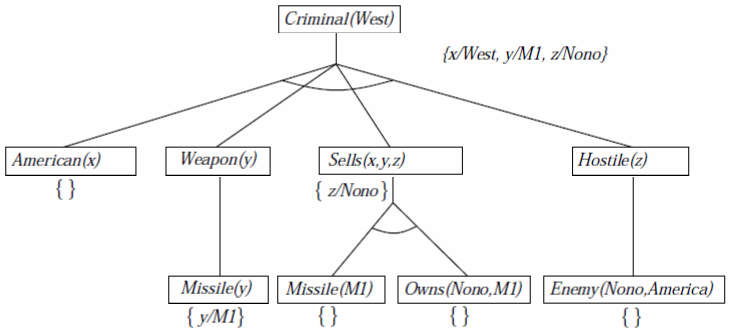
**function** FOL-BC-ASK($KB$, $goals$, $\theta$) **returns** a set of substitutions
  **inputs**: $KB$, a knowledge base
    $goals$, a list of conjuncts forming a query ($\theta$ already applied)
    $\theta$, the current substitution, initially the empty substitution $\{\ \}$
  **local variables**: $answers$, a set of substitutions, initially empty

  **if** $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST($\theta$, FIRST($goals$))
  **for each** sentence $r$ **in** $KB$
    where STANDARDIZE-APART($r$) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY($q$, $q'$) succeeds
   $new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$
   $answers \leftarrow$ FOL-BC-ASK($KB$, $new\_goals$, COMPOSE($\theta'$, $\theta$)) $\cup$ $answers$
  **return** $answers$





# Backward Chaining in FOL – Example Proof: Review



# Clausal Form Mnemonic – INSEUDOR: Review

- Implications Out (Replace with Disjunctive Clauses)
- Negations Inward (DeMorgan's Theorem)
- Standardize Variables Apart (Eliminate Duplicate Names)
- Existentials Out (Skolemize)
- Universals Made Implicit
- Distribute *And* Over *Or* (*i.e.*, Disjunctions Inward)
- Operators Made Implicit (Convert to List of Lists of Literals)
- Rename Variables (Independent Clauses)
- A Memonic for *Star Trek: The Next Generation* Fans

**Captain Picard:**

**I'll Notify Spock's Eminent Underground Dissidents On Romulus**

**I'll Notify Sarek's Eminent Underground Descendant On Romulus**

Adapted from: Nilsson and Genesereth (1987). *Logical Foundations of Artificial Intelligence.*
http://bit.ly/45Cmqq



# Skolemization: Eliminating Existential Quantifiers

$\exists x\ Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

$\exists\, k\ \ \frac{d}{dy}(k^y) = k^y$ becomes $\frac{d}{dy}(e^y) = e^y$

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"
$\forall x\ \ Person(x) \Rightarrow \exists y\ \ Heart(y) \wedge Has(x, y)$

Incorrect:
$\forall x\ \ Person(x) \Rightarrow Heart(H1) \wedge Has(x, H1)$

Correct:
$\forall x\ \ Person(x) \Rightarrow Heart(H(x)) \wedge Has(x, H(x))$
where $H$ is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

## Clausal Form (CNF) Conversion [1]: Review

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: $\neg\forall x, p\ \equiv \exists x\ \neg p$, $\neg\exists x, p\ \equiv \forall x\ \neg p$:

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





## Clausal Form (CNF) Conversion [2]: Review

3. Standardize variables: each quantifier should use a different one

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

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

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

5. Drop universal quantifiers:

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

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

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

# Resolution: Review

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 $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x)\\ Rich(Ken)\end{array}}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

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





# Refutation-Based Resolution Theorem Proving

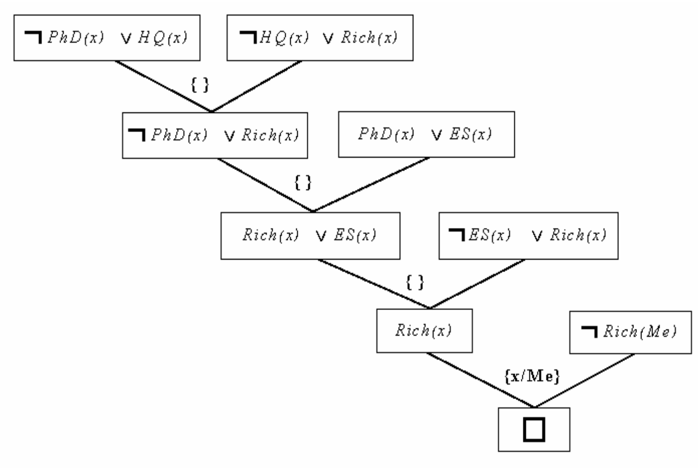
To prove $\alpha$:

- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

E.g., to prove $Rich(me)$, add $\neg Rich(me)$ to the CNF KB

$\neg PhD(x) \vee HighlyQualified(x)$
$PhD(x) \vee EarlyEarnings(x)$
$\neg HighlyQualified(x) \vee Rich(x)$
$\neg EarlyEarnings(x) \vee Rich(x)$

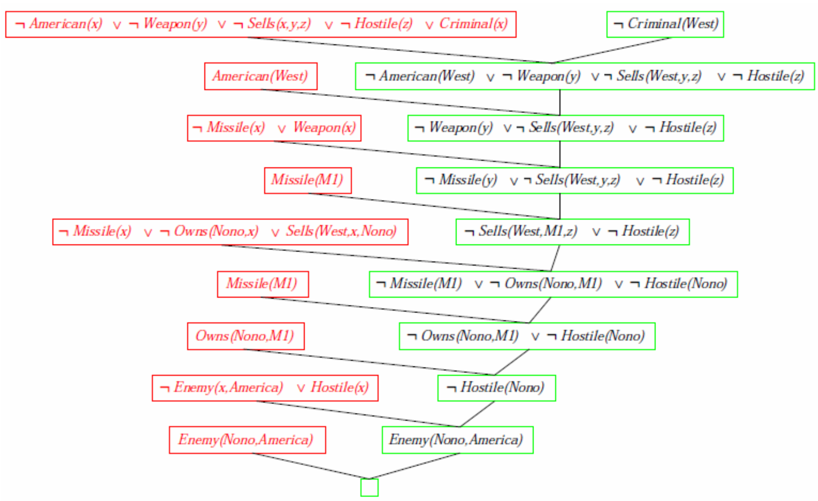
## Resolution Proof Example [1]




**Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.**



## Resolution Proof Example [2]




**Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.**

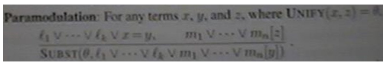
# Dealing with Equality in FOL: Demodulation & Paramodulation

- **Problem: How To Find Inference Rules for Sentences With =**
  - **Unification OK without it, but…**
  - ***A* = *B* doesn't force *P*(*A*) and *P*(*B*) to unify**
- **Solutions**
  - **Demodulation**
    - ⇨ **Generate substitution from equality term**
    - ⇨ **Additional sequent rule, Section 9.5 (p. 304) R&N 2[e]**

For any $x, y, z$ where $\text{UNIFY}(x, z) = \theta$ and $m_i$ contains z :

$$\frac{x = y, \quad m_1 \vee m_2 \vee \ldots \vee m_n[z]}{m_1 \vee m_2 \vee \ldots \vee m_n[\text{SUBST}(\theta, y)]}$$

  - **Paramodulation**
    - ⇨ **More powerful: *e.g.*, (*x* = *y*) ∨ P(x)**


**Paramodulation**: For any terms $x$, $y$, and $z$, where $\text{UNIFY}(x, z) = \theta$,

$$\frac{\ell_1 \vee \cdots \vee \ell_k \vee x = y, \qquad m_1 \vee \cdots \vee m_n[z]}{\text{SUBST}(\theta, \ell_1 \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_n[y])}.$$

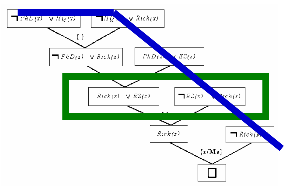


    - ⇨ **Generate substitution from WFF containing equality constraint**
    - ⇨ **Sequent rule sketch, Section 9.5 (p. 304) R&N 2[e]**
    - ⇨ **Full discussion in Nilsson & Genesereth**



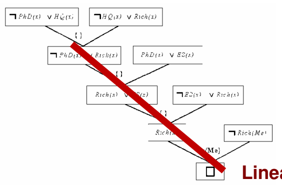
# Resolution Strategies [1]: Unit and Input Resolution

- **Unit Preference**
  - **Idea: Prefer inferences that produce shorter sentences**
  - **Compare: Occam's Razor**
  - **How? Prefer <u>unit clause</u> (*single-literal*) <u>resolvents</u> ( $\alpha \vee \beta$ with $\neg \beta \vee \alpha$ )**
  - **Reason: trying to produce a short sentence (⊥ ≡ True ⇒ False)**
- **Input Resolution**
  - **Idea: "diagonal" proof (proof "list" instead of proof tree)**
  - **Every resolution combines some input sentence with some other sentence**
  - **<u>Input sentence</u>: *in original KB or query***



# Resolution Strategies [2]: Linear Resolution and Set-of-Support

- **Linear Resolution**
  - Generalization of input resolution
  - Include any *ancestor in proof tree* to be used



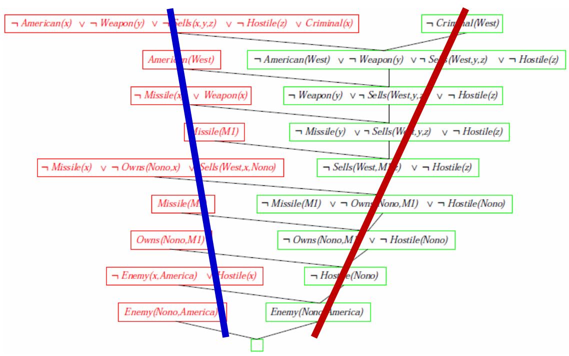
Linear resolutions

- **Set of Support (SoS)**
  - Idea: try to eliminate some potential resolutions
  - Prevention as opposed to cure
  - How?
    - ⇨ Maintain set SoS of resolution results
    - ⇨ Always take *one resolvent* from it
  - Caveat: need right choice for SoS to ensure completeness



# Resolution Strategies [3]: Subsumption

- **Subsumption**
  - Idea: eliminate sentences that sentences that are more specific than others
  - e.g., *P*(*x*) subsumes *P*(*A*)
- **Putting It All Together**



## Decision Problem [1]: FOL Validity / Unsatisfiability

- $L_{FOL\text{-}VALID}$ (written $L_{VALID}$): Language of Valid Sentences (Tautologies)
- Deciding Membership
  - Given: KB, $\alpha$
  - Decide: KB $\models \alpha$ ? (Is $\alpha$ valid? Is $\neg \alpha$ *contradictory*, *i.e.*, unsatisfiable?)
- Procedure
  - Test whether KB $\cup \{\neg \alpha\} \vdash_{RESOLUTION} \perp$
  - Answer <u>YES</u> if it does
- $L_{FOL\text{-}SAT}{}^{C}$ (written $L_{SAT}{}^{C}$) Language of Unsatisfiable Sentences
- Dual Problems $L_{VALID} \cong \overline{L_{SAT}} \Leftrightarrow \overline{L_{SAT}} \le L_{VALID}$ (direct proof) $\wedge$

  $L_{VALID} \le \overline{L_{SAT}}$ (refutation resolution)
- <u>Semi</u>-Decidable: $L_{VALID}$, $L_{SAT}{}^{C} \in$ RE \ REC ("Find A Contradiction")
  - Recursive enumerable but not recursive
  - Can return in finite steps and answer YES if $\alpha \in L_{VALID}$ or $\alpha \in L_{SAT}{}^{C}$
  - Can't return in finite steps and answer NO otherwise

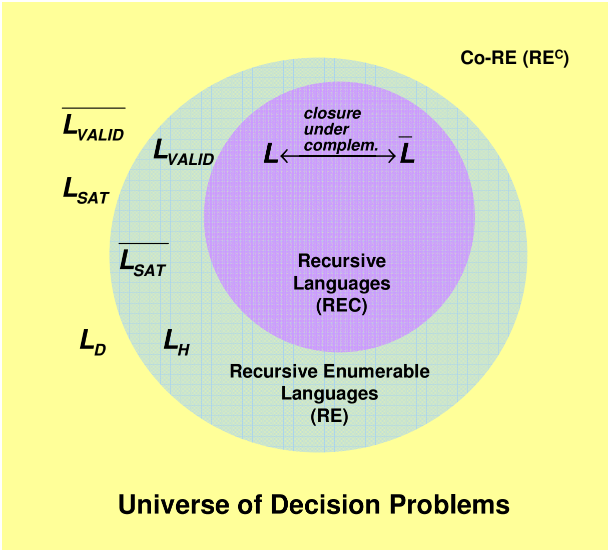


## Decision Problem [2]: FOL Non-Validity / Satisfiability

- $L_{FOL\text{-}VALID}{}^{C}$ (written $L_{VALID}{}^{C}$): Language of Non-Valid Sentences
- Deciding Membership
  - Given: KB, $\alpha$
  - Decide: KB $\not\models \alpha$ ? (Is there a counterexample to $\alpha$ ? *i.e.*, is $\neg \alpha$ satisfiable?)
- Procedure
  - Test whether KB $\cup \{\alpha\} \vdash_{RESOLUTION} \perp$
  - Answer <u>YES</u> if it does NOT
- $L_{FOL\text{-}SAT}$ (written $L_{SAT}$) Language of Satisfiable Sentences
- Dual Problems $\overline{L_{VALID}} \cong L_{SAT} \Leftrightarrow L_{SAT} \le \overline{L_{VALID}}$ (counterexample) $\wedge$

  $\overline{L_{VALID}} \le L_{SAT}$ (direct proof)
- <u>Un</u>decidable: $L_{VALID}{}^{C}$, $L_{SAT} \notin$ RE ("Find A Counterexample")
  - Not recursive enumerable
  - Can return in finite steps and answer NO if $\alpha \notin L_{VALID}{}^{C}$ or $\alpha \notin L_{SAT}$
  - Can't return in finite steps and answer YES otherwise

# Hilbert's Entscheidungsproblem and Other Decision Problems




$L_H$: Halting problem

$L_D$: Diagonal problem



# Logic Programming: Review

Sound bite: computation as inference on logical KBs

| | Logic programming | Ordinary programming |
|---|---|---|
| 1. | Identify problem | Identify problem |
| 2. | Assemble information | Assemble information |
| 3. | Tea break | Figure out solution |
| 4. | Encode information in KB | Program solution |
| 5. | Encode problem instance as facts | Encode problem instance as data |
| 6. | Ask queries | Apply program to data |
| 7. | Find false facts | Debug procedural errors |

Should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$ !

# Terminology

- **Resolution: Sound and Complete Inference Rule/Procedure for FOL**
  - Antecedent (*aka* precedent): sentences "above line" in sequent rule
  - Resolvent (*aka* consequent): sentences "below line" in sequent rule
- **Generalized Modus Ponens (GMP): Sequent Rule**
  - Forward chaining (FC) implementation: Rete algorithm
  - Backward chaining (BC) implementation
  - Fan-out (new facts) vs. fan-in (preconditions)
- **Clausal Form aka Conjunctive Normal Form (CNF)**
  - Canonical (standard) form for theorem proving by resolution, FC, BC
  - INSEUDOR procedure
- **Decision Problems: True-False for Membership in Formal Language**
  - Recursive: decidable
  - Recursive enumerable: decidable or semi-decidable
  - Semi-decidable: can answer YES when true but not necessarily NO
  - Undecidable: can't necessarily answer YES when true
  - Hilbert's $10^{th}$ problem (*Entscheidungsproblem*): mathematical truth



# Summary Points

- **Last Class: GMP, Unification, Forward and Backward Chaining**
  - Generalized Modus Ponens (GMP)
    - ⇨ Sound, complete rule for first-order inference (reasoning in FOL)
    - ⇨ Requires pattern matching by unification
  - Unification: matches well-formed formulas (WFFs), atoms, terms
- **Resolution: Sound and Complete Inference Rule/Procedure for FOL**
  - Proof example in detail
  - Paramodulation and demodulation
  - Resolution strategies: unit, linear, input, set of support
  - FOL and computability: complements and duals
    - ⇨ Semidecidability ($\in$ RE \ REC) of validity, unsatisfiability
    - ⇨ Undecidability of ($\notin$ RE) of non-validity, satisfiability
  - Decision problems
  - Relation to computability, formal languages (Type-0 Chomsky language)
- **Read About: Russell's Paradox**
- **Next: Prolog (Briefly); Knowledge Engineering, Ontology Intro**