First-order logic.

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Administration

• PS-4:
  – Due on Thursday, October 7, 2004
• Midterm:
  – October 19, 2004 during the class
  – Closed book
  – Covers:
    Search and Knowledge Representation
Logic

Logic is defined by:

- **A set of sentences**
  - A sentence is constructed from a set of primitives according to syntax rules.

- **A set of interpretations**
  - An interpretation gives a semantic to primitives. It associates primitives with objects, values in the real world.

- **The valuation (meaning) function** $V$
  - Assigns a truth value to a given sentence under some interpretation
  
  $V : \text{sentence} \times \text{interpretation} \rightarrow \{ \text{True} , \text{False} \}$

First-order logic. Syntax.

**Term** - syntactic entity for representing objects

**Terms in FOL:**

- **Constant symbols:** represent specific objects
  - E.g. *John, France, car89*

- **Variables:** represent objects of a certain type (type = domain of discourse)
  - E.g. *x, y, z*

- **Functions** applied to one or more terms
  - E.g. *father-of (John)*
    
    $\text{father-of(father-of(John))}$
First order logic. Syntax.

Sentences in FOL:
• Atomic sentences:
  – A predicate symbol applied to 0 or more terms
    Examples:
    \[ \text{Red(car12)}, \]
    \[ \text{Sister(Amy, Jane)}; \]
    \[ \text{Manager(father-of(John))}; \]
  – \( t_1 = t_2 \) equivalence of terms
    Example:
    \[ \text{John = father-of(Peter)} \]

First order logic. Syntax.

Sentences in FOL:
• Complex sentences:
  • Assume \( \phi, \psi \) are sentences in FOL. Then:
    \[ (\phi \land \psi) (\phi \lor \psi) (\phi \Rightarrow \psi) (\phi \Leftrightarrow \psi) \neg \psi \]
    and
    \[ \forall x \phi \quad \exists y \phi \]
    are sentences

Symbols \( \exists, \forall \)
- stand for the existential and the universal quantifier
Semantics. Interpretation.

An interpretation $I$ is defined by a mapping of symbols to the domain of discourse $D$ or relations on $D$

- **domain of discourse**: a set of objects in the world we represent and refer to;

**An interpretation $I$ maps:**

- Constant symbols to objects in $D$
  $I(John) = \Box$
- Predicate symbols to relations, properties on $D$
  $I(brother) = \{ \langle \Box \Box \Box \rangle; \langle \Box \Box \Box \rangle; \ldots \}$
- Function symbols to functional relations on $D$
  $I(father-of) = \{ \langle \Box \Box \Box \rangle \rightarrow \Box; \langle \Box \Box \Box \rangle \rightarrow \Box; \ldots \}$

Semantics of sentences.

**Meaning (evaluation) function:**

$V: \text{sentence} \times \text{interpretation} \rightarrow \{ \text{True}, \text{False} \}$

A predicate $\text{predicate}(\text{term-1, term-2, term-3, term-n})$ is true for the interpretation $I$, iff the objects referred to by $\text{term-1, term-2, term-3, term-n}$ are in the relation referred to by $\text{predicate}$

$I(John) = \Box$ \quad $I(Paul) = \Box$

$I(brother) = \{ \langle \Box \Box \Box \rangle; \langle \Box \Box \Box \rangle; \ldots \}$

$\text{brother}(John, Paul) = \langle \Box \Box \Box \rangle$ \quad in $I(brother)$

$V(\text{brother}(John, Paul), I) = \text{True}$
Semantics of sentences.

- **Equality**
  \[ V(\text{term-1} = \text{term-2}, I) = \text{True} \]
  Iff \[ I(\text{term-1}) = I(\text{term-2}) \]

- **Boolean expressions:** standard
  
  E.g. \[ V(\text{sentence-1} \lor \text{sentence-2}, I) = \text{True} \]
  Iff \[ V(\text{sentence-1}, I) = \text{True} \] or \[ V(\text{sentence-2}, I) = \text{True} \]

- **Quantifications**
  
  \[ V(\forall x \ \phi, I) = \text{True} \]
  Iff for all \( d \in D \) \[ V(\phi, I[x/d]) = \text{True} \]
  
  \[ V(\exists x \ \phi, I) = \text{True} \]
  Iff there is a \( d \in D \), s.t. \[ V(\phi, I[x/d]) = \text{True} \]

Order of quantifiers

- **Order of quantifiers of the same type does not matter**
  For all \( x \) and \( y \), if \( x \) is a parent of \( y \) then \( y \) is a child of \( x \)
  \[ \forall x, \ y \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x) \]
  \[ \forall y, \ x \ \text{parent} \ (x, y) \Rightarrow \text{child} \ (y, x) \]

- **Order of different quantifiers changes the meaning**
  \[ \forall x \exists y \ \text{loves} \ (x, y) \]
Order of quantifiers

- **Order of quantifiers of the same type does not matter**
  
  *For all x and y, if x is a parent of y then y is a child of x*
  
  \[
  \forall x, y \text{ parent } (x, y) \Rightarrow \text{ child } (y, x) \\
  \forall y, x \text{ parent } (x, y) \Rightarrow \text{ child } (y, x)
  \]

- **Order of different quantifiers changes the meaning**
  
  \[
  \forall x \exists y \text{ loves } (x, y)
  \\
  \text{Everybody loves somebody}
  \\
  \exists y \forall x \text{ loves } (x, y)
  \text{There is someone who is loved by everyone}
  \]
Connections between quantifiers

Everyone likes ice cream

\( \forall x \text{ likes } (x, \text{IceCream} ) \)

Is it possible to convey the same meaning using an existential quantifier?

There is no one who does not like ice cream

\( \neg \exists x \neg \text{ likes } (x, \text{IceCream} ) \)

A universal quantifier in the sentence can be expressed using an existential quantifier !!!

Connections between quantifiers

Someone likes ice cream

\( \exists x \text{ likes } (x, \text{IceCream} ) \)

Is it possible to convey the same meaning using a universal quantifier?

Not everybody does not like ice cream

\( \neg \forall x \neg \text{ likes } (x, \text{IceCream} ) \)

An existential quantifier in the sentence can be expressed using a universal quantifier !!!
Representing knowledge in FOL

Example:

**Kinship domain**

- **Objects:** people
  
  \( \text{John, Mary, Jane, …} \)

- **Properties:** gender
  
  \( \text{Male} \ (x), \text{Female} \ (x) \)

- **Relations:** parenthood, brotherhood, marriage
  
  \( \text{Parent} \ (x, y), \text{Brother} \ (x, y), \text{Spouse} \ (x, y) \)

- **Functions:** mother-of (one for each person \( x \))
  
  \( \text{MotherOf} \ (x) \)

---

Kinship domain in FOL

**Relations between predicates and functions:** write down what we know about them; how relate to each other.

- Male and female are disjoint categories
  
  \( \forall x \ \text{Male} \ (x) \iff \neg \text{Female} \ (x) \)

- Parent and child relations are inverse
  
  \( \forall x, y \ \text{Parent} \ (x, y) \iff \text{Child} \ (y, x) \)

- A grandparent is a parent of parent
  
  \( \forall g, c \ \text{Grandparent} (g, c) \iff \exists p \ \text{Parent} (g, p) \land \text{Parent} (p, c) \)

- A sibling is another child of one’s parents
  
  \( \forall x, y \ \text{Sibling} \ (x, y) \iff (x \neq y) \land \exists p \ \text{Parent} (p, x) \land \text{Parent} (p, y) \)

- And so on ….
Logical inference in FOL

Logical inference problem:
• Given a knowledge base KB (a set of sentences) and a sentence $\alpha$, does the KB semantically entail $\alpha$?

$$KB \models \alpha ?$$

In other words: In all interpretations in which sentences in the KB are true, is also $\alpha$ true?

Logical inference problem in the first-order logic is **undecidable** !!! No procedure that can decide the entailment for all possible input sentences in a finite number of steps.
Logical inference problem in the Propositional logic

Computational procedures that answer:

\[ KB \models \alpha \ ? \]

Three approaches:
- Truth-table approach
- Inference rules
- Conversion to the inverse SAT problem
  - Resolution-refutation

Inference in FOL: Truth table approach

- Is the Truth-table approach a viable approach for the FOL?
  - NO!
  - Why?
  - It would require us to enumerate and list all possible interpretations I
  - I = (assignments of symbols to objects, predicates to relations and functions to relational mappings)
  - Simply there are too many interpretations
Inference in FOL: Inference rules

• Is the Inference rule approach a viable approach for the FOL?

  • Yes.

• The inference rules represent sound inference patterns one can apply to sentences in the KB

• What is derived follows from the KB

• Caveat: we need to add rules for handling quantifiers

Inference rules

• **Inference rules from the propositional logic:**
  – Modus ponens
    \[ \begin{align*}
    A \rightarrow B, & \quad A \\
    \hline
    & \quad B
    \end{align*} \]
  – Resolution
    \[ \begin{align*}
    A \lor B, \quad \neg B \lor C \\
    \hline
    & \quad A \lor C
    \end{align*} \]
  – and others: And-introduction, And-elimination, Or-introduction, Negation elimination

• **Additional inference rules** are needed for sentences with quantifiers and variables
  – Must involve variable substitutions
**Sentences with variables**

First-order logic sentences can include variables.

- **Variable** is:
  - **Bound** – if it is in the scope of some quantifier
    \[ \forall x \ P(x) \]
  - **Free** – if it is not bound,
    \[ \exists x \ P(y) \land Q(x) \quad y \text{ is free} \]

- **Sentence** (formula) is:
  - **Closed** – if it has no free variables
    \[ \forall y \exists x \ P(y) \implies Q(x) \]
  - **Open** – if it is not closed
  - **Ground** – if it does not have any variables
    \[ Likes(John, Jane) \]

---

**Variable substitutions**

- Variables in the sentences can be substituted with terms. (terms = constants, variables, functions)
- **Substitution:**
  - Is represented by a mapping from variables to terms
    \[ \{x_1 / t_1, x_2 / t_2, \ldots \} \]
  - Application of the substitution to sentences
    \[
    SUBST(\{x / Sam, y / Pam\}, Likes(x, y)) = Likes(Sam, Pam)
    \]
    \[
    SUBST(\{x / z, y / fatherof(John)\}, Likes(x, y)) = Likes(z, fatherof(John))
    \]
Inference rules for quantifiers

• **Universal elimination**
\[
\frac{\forall x \phi(x)}{\phi(a)} \quad a - \text{is a constant symbol}
\]
– Substitutes a variable with a constant symbol
\[\forall x \text{ Likes}(x, \text{IceCream}) \quad \text{Likes}(\text{Ben, IceCream})\]

• **Existential elimination.**
\[
\frac{\exists x \phi(x)}{\phi(a)}
\]
– Substitutes a variable with a constant symbol that does not appear elsewhere in the KB
\[\exists x \text{ Kill}(x, \text{Victim}) \quad \text{Kill(\text{Murderer, Victim})}\]

Inference rules for quantifiers

• **Universal instantiation (introduction)**
\[
\frac{\phi}{\forall x \phi} \quad x - \text{is not free in } \phi
\]
– Introduces a universal variable which does not affect \( \phi \) or its assumptions
\[\text{Sister(\text{Amy, Jane})} \quad \forall x \text{ Sister(\text{Amy, Jane})}\]

• **Existential instantiation (introduction)**
\[
\frac{\phi(a)}{\exists x \phi(x)} \quad a - \text{is a ground term in } \phi
\]
– Substitutes a ground term in the sentence with a variable and an existential statement
\[\text{Likes}(\text{Ben, IceCream}) \quad \exists x \text{ Likes}(x, \text{IceCream})\]
Unification

• **Problem in inference**: Universal elimination gives many opportunities for substituting variables with ground terms
  \[ \forall x \phi(x) \quad \frac{\phi(a)}{a} \quad a \text{ - is a constant symbol} \]

• **Solution**: Try substitutions that may help
  – Use substitutions of “similar” sentences in KB

• **Unification** – takes two similar sentences and computes the substitution that makes them look the same, if it exists

  \[ \text{UNIFY} (p, q) = \sigma \quad \text{s.t.} \quad \text{SUBST}(\sigma, p) = \text{SUBST}(\sigma, q) \]

---

Unification. Examples.

• **Unification**:
  \[ \text{UNIFY} (p, q) = \sigma \quad \text{s.t.} \quad \text{SUBST}(\sigma, p) = \text{SUBST}(\sigma, q) \]

• **Examples**:
  \[ \text{UNIFY} (\text{Knows}(John, x), \text{Knows}(John, Jane)) = \{x / Jane\} \]
  \[ \text{UNIFY}(\text{Knows}(John, x), \text{Knows}(y, Ann)) = \{x / Ann, y / John\} \]
  \[ \text{UNIFY} (\text{Knows} (John, x), \text{Knows} (y, \text{MotherOf} (y)))
      = \{x / \text{MotherOf}(John), y / John\} \]
  \[ \text{UNIFY} (\text{Knows}(John, x), \text{Knows}(x, Elizabeth)) = \text{fail} \]
Generalized inference rules.

- **Use substitutions that let us make inferences**
  
  **Example:** Modus Ponens
  
- **If there exists a substitution** $\sigma$ **such that**

  $$\text{SUBST } (\sigma, A_i) = \text{SUBST } (\sigma, A'_i) \quad \text{for all } i=1,2,n$$

  $$A_1 \land A_2 \land \ldots A_n \Rightarrow B, \quad A'_1, A'_2, \ldots A'_n$$

  $$\text{SUBST } (\sigma, B)$$

- Substitution that satisfies the generalized inference rule can be built via unification process
- Advantage of the generalized rules: they are focused
  - only substitutions that allow the inferences to proceed

Resolution inference rule

- **Recall:** Resolution inference rule is sound and complete (refutation-complete) for the *propositional logic* and CNF

  $$A \lor B, \quad \lnot A \lor C$$

  $$\frac{}{B \lor C}$$

- **Generalized resolution rule is sound and refutation complete** for the first-order logic and CNF w/o equalities (if unsatisfiable the resolution will find the contradiction)

  $$\sigma = \text{UNIFY } (\phi_i, \lnot \psi_j) \neq \text{fail}$$

  $$\phi_1 \lor \phi_2 \ldots \lor \phi_k, \quad \psi_1 \lor \psi_2 \lor \ldots \psi_n$$

  $$\text{SUBST}(\sigma, \phi_1 \lor \ldots \lor \phi_{i-1} \lor \phi_{i+1} \ldots \lor \phi_k \lor \psi_1 \lor \ldots \lor \psi_{j-1} \lor \psi_{j+1} \ldots \psi_n)$$

  **Example:**

  $$P(x) \lor Q(x), \quad \lnot Q(\text{John}) \lor S(y)$$

  $$\frac{}{P(\text{John}) \lor S(y)}$$
Inference with resolution rule

- **Proof by refutation:**
  - Prove that $KB, \neg \alpha$ is unsatisfiable
  - resolution is refutation-complete

- **Main procedure (steps):**
  1. Convert $KB, \neg \alpha$ to CNF with ground terms and universal variables only
  2. Apply repeatedly the resolution rule while keeping track and consistency of substitutions
  3. Stop when empty set (contradiction) is derived or no more new resolvents (conclusions) follow

---

Conversion to CNF

1. Eliminate implications, equivalences
   \[(p \Rightarrow q) \rightarrow (\neg p \lor q)\]

2. Move negations inside (DeMorgan’s Laws, double negation)
   \[-(p \land q) \rightarrow \neg p \lor \neg q\]
   \[-(p \lor q) \rightarrow \neg p \land \neg q\]
   \[-\forall x \ p \rightarrow \exists x \ \neg p\]
   \[-\exists x \ p \rightarrow \forall x \ \neg p\]
   \[\neg \neg p \rightarrow p\]

3. Standardize variables (rename duplicate variables)
   \[(\forall x \ P(x)) \lor (\exists x \ Q(x)) \rightarrow (\forall x \ P(x)) \lor (\exists y \ Q(y))\]
Conversion to CNF

4. **Move all quantifiers left** (no invalid capture possible)
   \[(\forall x \ P(x)) \lor (\exists y \ Q(y)) \rightarrow \forall x \ \exists y \ P(x) \lor Q(y)\]

5. **Skolemization** (removal of existential quantifiers through elimination)
   - If no universal quantifier occurs before the existential quantifier, replace the variable with a new constant symbol
     \[\exists y \ P(A) \lor Q(y) \rightarrow P(A) \lor Q(B)\]
   - If a universal quantifier precede the existential quantifier replace the variable with a function of the “universal” variable
     \[\forall x \ \exists y \ P(x) \lor Q(y) \rightarrow \forall x \ P(x) \lor Q(F(x))\]
     \[F(x) \quad - \text{a Skolem function}\]

---

Conversion to CNF

6. **Drop universal quantifiers** (all variables are universally quantified)
   \[\forall x \ P(x) \lor Q(F(x)) \rightarrow P(x) \lor Q(F(x))\]

7. **Convert to CNF using the distributive laws**
   \[p \lor (q \land r) \rightarrow (p \lor q) \land (p \lor r)\]

   The result is a CNF with variables, constants, functions
Resolution example

KB
\[ \neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \neg R(z) \lor S(z), \ \neg S(A) \]

\[ \neg \alpha \]

CS 2710 Foundations of AI
Resolution example

\[
\begin{align*}
\text{KB} & \quad \neg \alpha \\
\neg P(w) \lor Q(w), \quad & \neg \neg Q(y) \lor S(y), \quad P(x) \lor R(x), \quad \neg \neg R(z) \lor S(z), \quad \neg \neg S(A) \\
\end{align*}
\]

\[
\begin{align*}
\neg P(w) \lor S(w) & \quad \{x / w\} \\
S(w) \lor R(w) & \quad \{y / w\} \\
\end{align*}
\]

CS 2710 Foundations of AI

Resolution example

\[
\begin{align*}
\text{KB} & \quad \neg \alpha \\
\neg P(w) \lor Q(w), \quad & \neg \neg Q(y) \lor S(y), \quad P(x) \lor R(x), \quad \neg \neg R(z) \lor S(z), \quad \neg \neg S(A) \\
\end{align*}
\]

\[
\begin{align*}
\neg P(w) \lor S(w) & \quad \{x / w\} \\
S(w) \lor R(w) & \quad \{y / w\} \\
z/w & \quad \{z / w\} \\
S(w) & \quad \{\_ / \_\} \\
\end{align*}
\]

CS 2710 Foundations of AI
Resolution example

\[ \neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \neg R(z) \lor S(z), \neg S(A) \]

\[ \{y / w\} \]

\[ \neg P(w) \lor S(w) \]

\[ \{x / w\} \]

\[ S(w) \lor R(w) \]

\[ \{z / w\} \]

\[ S(w) \{w / A\} \]

\[ KB \models \alpha \]

Contradiction

Dealing with equality

- Resolution works for first-order logic without equalities
- To incorporate equalities we need an additional inference rule
- Demodulation rule
  \[ \sigma = \text{UNIFY} (\phi_1, t_1) \neq \text{fail} \]
  \[
  \phi_1 \lor \phi_2 \ldots \lor \phi_k, \ t_1 = t_2
  \]
  \[
  \frac{\text{SUBST}\{\text{SUBST}(\sigma, t_1) / \text{SUBST}(\sigma, t_2)\}, \phi_1 \lor \ldots \lor \phi_{i-1} \lor \phi_{i+1} \ldots \lor \phi_k}{\text{SUBST}(\{t_1 / t_2\})}
  \]
- Example:
  \[
  P(f(a)), f(x) = x
  \]
  \[ P(a) \]
- Paramodulation rule: more powerful
- Resolution+paramodulation give a refutation-complete proof theory for FOL
Sentences in Horn normal form

• **Horn normal form (HNF) in the propositional logic**
  – a special type of clause with at most one positive literal
    \[ (A \lor \neg B) \land (\neg A \lor \neg C \lor D) \]

    Typically written as: \((B \Rightarrow A) \land ((A \land C) \Rightarrow D)\)

• A clause with one literal, e.g. \(A\), is also called a **fact**.

• A clause representing an implication (with a conjunction of positive literals in antecedent and one positive literal in consequent), is also called a **rule**.

• **Generalized Modus ponens:**
  – is the **complete inference rule** for KBs in the Horn normal form. **Not all KBs are convertible to HNF !!!**

Horn normal form in FOL

**First-order logic (FOL)**
  – adds variables and quantifiers, works with terms

**Generalized modus ponens rule:**

\[
\sigma = \text{a substitution s.t. } \forall i \text{ SUBST}(\sigma, \phi_i) = \text{SUBST}(\sigma, \phi_1)
\]

\[
\phi_1', \phi_2', \ldots, \phi_n', \phi_1 \land \phi_2 \land \ldots \land \phi_n \Rightarrow \tau \\
\text{SUBST } (\sigma, \tau)
\]

**Generalized modus ponens:**

• is **complete** for the KBs with sentences in Horn form;

• Not all first-order logic sentences can be expressed in this form.
Forward and backward chaining

Two inference procedures based on modus ponens for Horn KBs:

- **Forward chaining**
  
  **Idea:** Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.
  
  **Typical usage:** If we want to infer all sentences entailed by the existing KB.

- **Backward chaining (goal reduction)**
  
  **Idea:** To prove the fact that appears in the conclusion of a rule prove the premises of the rule. Continue recursively.
  
  **Typical usage:** If we want to prove that the target (goal) sentence $\alpha$ is entailed by the existing KB.

Both procedures are complete for KBs in Horn form !!!

Forward chaining example

- **Forward chaining**
  
  **Idea:** Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied
  
  Assume the KB with the following rules:

<table>
<thead>
<tr>
<th>KB:</th>
<th>Rule Number</th>
<th>Rule Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1:</td>
<td>$Steamboat (x) \land Sailboat (y) \Rightarrow Faster (x, y)$</td>
<td></td>
</tr>
<tr>
<td>R2:</td>
<td>$Sailboat (y) \land RowBoat (z) \Rightarrow Faster (y, z)$</td>
<td></td>
</tr>
<tr>
<td>R3:</td>
<td>$Faster (x, y) \land Faster (y, z) \Rightarrow Faster (x, z)$</td>
<td></td>
</tr>
</tbody>
</table>

| F1: | $Steamboat (Titanic)$ |
| F2: | $Sailboat (Mistral)$ |
| F3: | $RowBoat(PondArrow)$ |

Theorem: $Faster (Titanic, PondArrow)$
Forward chaining example

KB:  
R1:  Steamboat (x) ∧ Sailboat (y) ⇒ Faster (x, y)  
R2:  Sailboat (y) ∧ RowBoat (z) ⇒ Faster (y, z)  
R3:  Faster (x, y) ∧ Faster (y, z) ⇒ Faster (x, z)  

F1:  Steamboat (Titanic)  
F2:  Sailboat (Mistral)  
F3:  RowBoat(PondArrow)  

?  

Rule R1 is satisfied:
F4:  Faster(Titanic,Mistral)
Forward chaining example

KB:

R1:  \( \text{Steamboat} (x) \land \text{Sailboat} (y) \Rightarrow \text{Faster} (x, y) \)
R2:  \( \text{Sailboat} (y) \land \text{RowBoat} (z) \Rightarrow \text{Faster} (y, z) \)
R3:  \( \text{Faster} (x, y) \land \text{Faster} (y, z) \Rightarrow \text{Faster} (x, z) \)

F1:  \( \text{Steamboat} (\text{Titanic}) \)
F2:  \( \text{Sailboat} (\text{Mistral}) \)
F3:  \( \text{RowBoat}(\text{PondArrow}) \)

Rule R1 is satisfied:
F4:  \( \text{Faster}(\text{Titanic}, \text{Mistral}) \)

Rule R2 is satisfied:
F5:  \( \text{Faster}(\text{Mistral}, \text{PondArrow}) \)

Rule R3 is satisfied:
F6:  \( \text{Faster}(\text{Titanic}, \text{PondArrow}) \)

Forward chaining example

KB:

R1:  \( \text{Steamboat} (x) \land \text{Sailboat} (y) \Rightarrow \text{Faster} (x, y) \)
R2:  \( \text{Sailboat} (y) \land \text{RowBoat} (z) \Rightarrow \text{Faster} (y, z) \)
R3:  \( \text{Faster} (x, y) \land \text{Faster} (y, z) \Rightarrow \text{Faster} (x, z) \)

F1:  \( \text{Steamboat} (\text{Titanic}) \)
F2:  \( \text{Sailboat} (\text{Mistral}) \)
F3:  \( \text{RowBoat}(\text{PondArrow}) \)

Rule R1 is satisfied:
F4:  \( \text{Faster}(\text{Titanic}, \text{Mistral}) \)

Rule R2 is satisfied:
F5:  \( \text{Faster}(\text{Mistral}, \text{PondArrow}) \)

Rule R3 is satisfied:
F6:  \( \text{Faster}(\text{Titanic}, \text{PondArrow}) \)
Backward chaining example

- **Backward chaining (goal reduction)**
  
  **Idea:** To prove the fact that appears in the conclusion of a rule prove the antecedents (if part) of the rule & repeat recursively.

**KB:**

- **R1:** \(\text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y)\)
- **R2:** \(\text{Sailboat}(y) \land \text{RowBoat}(z) \Rightarrow \text{Faster}(y, z)\)
- **R3:** \(\text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z)\)

**F1:** \(\text{Steamboat}(\text{Titanic})\)

**F2:** \(\text{Sailboat}(\text{Mistral})\)

**F3:** \(\text{RowBoat}(\text{PondArrow})\)

**Theorem:** \(\text{Faster}(\text{Titanic}, \text{PondArrow})\)

---

**Backward chaining example**

\[
\begin{align*}
\text{Faster}(\text{Titanic}, \text{PondArrow}) & \quad \text{F1: Steamboat (Titanic)} \\
\text{Steamboat}(\text{Titanic}) & \quad \text{F2: Sailboat (Mistral)} \\
\text{Sailboat}(\text{PondArrow}) & \quad \text{F3: RowBoat(PondArrow)}
\end{align*}
\]

\[
\begin{align*}
\text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y) & \quad \text{R1:}\ \\\text{Steamboat}(\text{Titanic}) \\
\text{Faster}(\text{Titanic}, \text{PondArrow}) & \quad \text{R2:}\ \\\text{Sailboat}(\text{Mistral}) \\
\{x / \text{Titanic}, y / \text{PondArrow}\} & \quad \text{R3:}\ \\\text{RowBoat(PondArrow)}
\end{align*}
\]
Backward chaining example

Faster(Titanic, PondArrow)

Steamboat(Titanic)

\(\checkmark\)

Sailboat(Titanic)

\(\times\)

RowBoat(PondArrow)

\(\checkmark\)

\(\text{Faster (y, z)} \Rightarrow Faster (Titanic, PondArrow)\)

\(\{y / Titanic, z / PondArrow\}\)

CS 2710 Foundations of AI
Backward chaining

- The search tree: **AND/OR tree**
- Special search algorithms exits (including heuristics): AO, AO*

\[
\begin{align*}
\text{Faster}(\text{Titanic}, \text{PondArrow}) & \quad \text{R3} \\
\text{Steamboat}(\text{Titanic}) & \quad \text{R1} \\
\text{Sailboat}(\text{Titanic}) & \quad \text{R2} \\
\text{RowBoat}(\text{PondArrow}) & \\
\text{Faster}(\text{Titanic}, y) & \\
\text{Steamboat}(\text{Titanic}) & \quad \text{R1} \\
\text{Sailboat}(\text{Mistral}) & \quad \text{R2} \\
\text{RowBoat}(\text{PondArrow}) & \\
\text{Faster}(y, \text{PondArrow}) & \\
\text{Sailboat}(\text{Mistral}) & \quad \text{R1} \\
\text{RowBoat}(\text{PondArrow}) & \\
\text{Faster}(y, \text{PondArrow}) \\
\text{Sailboat}(\text{Mistral}) & \quad \text{R2} \\
\text{RowBoat}(\text{PondArrow}) & \\
\text{Faster}(y, \text{PondArrow})
\end{align*}
\]