

CS 1571 Introduction to AI

Lecture 19

Inference in first-order logic

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

Logical inference in FOL

Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence α , does the KB semantically entail α ?

$$KB \models \alpha \quad ?$$

In other words: In all interpretations in which sentences in the KB are true, is also α 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 rules

- Inference rules from the propositional logic:

- Modus ponens

$$\frac{A \Rightarrow B, \quad A}{B}$$

- Resolution

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

- 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

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

$$\{x_1 / t_1, x_2 / t_2, \dots\}$$

- 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

$$\forall x Likes(x, IceCream) \quad Likes(Ben, IceCream)$$

- Existential elimination.**

$$\frac{\exists x \phi(x)}{\phi(a)}$$

- Substitutes a variable with a constant symbol that does not appear elsewhere in the KB

$$\exists x Kill(x, Victim) \quad Kill(Murderer, Victim)$$

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 ϕ or its assumptions

$$Sister(Amy, Jane) \quad \forall x Sister(Amy, Jane)$$

- **Existential instantiation (introduction)**

$$\frac{\phi(a)}{\exists x \phi(x)} \quad \begin{array}{l} a - \text{is a ground term in } \phi \\ x - \text{is not free in } \phi \end{array}$$

- Substitutes a ground term in the sentence with a variable and an existential statement

$$Likes(Ben, IceCream) \quad \exists x Likes(x, IceCream)$$

Unification

- **Problem in inference:** Universal elimination gives many opportunities for substituting variables with ground terms

$$\frac{\forall x \phi(x)}{\phi(a)} \quad a - \text{is a constant symbol}$$

- **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

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

Unification. Examples.

- **Unification:**

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

- **Examples:**

$$UNIFY(Knows(John, x), Knows(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x / Ann, y / John\}$$

$$UNIFY(Knows(John, x), Knows(y, MotherOf(y))) \\ = \{x / MotherOf(John), y / John\}$$

$$UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail$$

Generalized inference rules.

- **Use substitutions that let us make inferences**

Example: Modus Ponens

- **If there exists a substitution σ such that**

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

$$\frac{A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow B, \quad A_1', A_2', \dots, A_n'}{SUBST(\sigma, B)}$$

- Substitution that satisfies the generalized inference rule can be build 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

$$\frac{A \vee B, \quad \neg A \vee C}{B \vee C}$$

- **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)

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

$$\frac{\phi_1 \vee \phi_2 \dots \vee \phi_k, \quad \psi_1 \vee \psi_2 \vee \dots \vee \psi_n}{\text{SUBST}(\sigma, \phi_1 \vee \dots \vee \phi_{i-1} \vee \phi_{i+1} \dots \vee \phi_k \vee \psi_1 \vee \dots \vee \psi_{j-1} \vee \psi_{j+1} \dots \vee \psi_n)}$$

Example:
$$\frac{P(x) \vee Q(x), \quad \neg Q(\text{John}) \vee S(y)}{?}$$

Resolution inference rule

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

$$\frac{A \vee B, \quad \neg A \vee C}{B \vee C}$$

- **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)

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

$$\frac{\phi_1 \vee \phi_2 \dots \vee \phi_k, \quad \psi_1 \vee \psi_2 \vee \dots \vee \psi_n}{\text{SUBST}(\sigma, \phi_1 \vee \dots \vee \phi_{i-1} \vee \phi_{i+1} \dots \vee \phi_k \vee \psi_1 \vee \dots \vee \psi_{j-1} \vee \psi_{j+1} \dots \vee \psi_n)}$$

Example:
$$\frac{P(x) \vee Q(x), \quad \neg Q(\text{John}) \vee S(y)}{P(\text{John}) \vee S(y)}$$

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**

$$(p \Rightarrow q) \rightarrow (\neg p \vee q)$$

2. **Move negations inside** (DeMorgan's Laws, double negation)

$$\begin{aligned}\neg(p \wedge q) &\rightarrow \neg p \vee \neg q & \neg \forall x p &\rightarrow \exists x \neg p \\ \neg(p \vee q) &\rightarrow \neg p \wedge \neg q & \neg \exists x p &\rightarrow \forall x \neg p \\ & & \neg \neg p &\rightarrow p\end{aligned}$$

3. **Standardize variables** (rename duplicate variables)

$$(\forall x P(x)) \vee (\exists x Q(x)) \rightarrow (\forall x P(x)) \vee (\exists y Q(y))$$

4. **Move all quantifiers left** (no invalid capture possible)

$$(\forall x P(x)) \vee (\exists y Q(y)) \rightarrow \forall x \exists y P(x) \vee Q(y)$$

Conversion to CNF

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**

$$\exists y P(A) \vee Q(y) \rightarrow P(A) \vee Q(B)$$

- If a universal quantifier precedes the existential quantifier replace the variable with a function of the “universal” variable

$$\forall x \exists y P(x) \vee Q(y) \rightarrow \forall x P(x) \vee Q(F(x))$$

$F(x)$ - **a special function**
- **called Skolem function**

Conversion to CNF

6. Drop universal quantifiers (all variables are universally quantified)

$$\forall x P(x) \vee Q(F(x)) \rightarrow P(x) \vee Q(F(x))$$

7. Convert to CNF using the distributive laws

$$p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)$$

The result is a CNF with variables, constants, functions

Resolution example

KB

$\neg \alpha$

$\neg P(w) \vee Q(w), \neg Q(y) \vee S(y), P(x) \vee R(x), \neg R(z) \vee S(z), \neg S(A)$

Resolution example

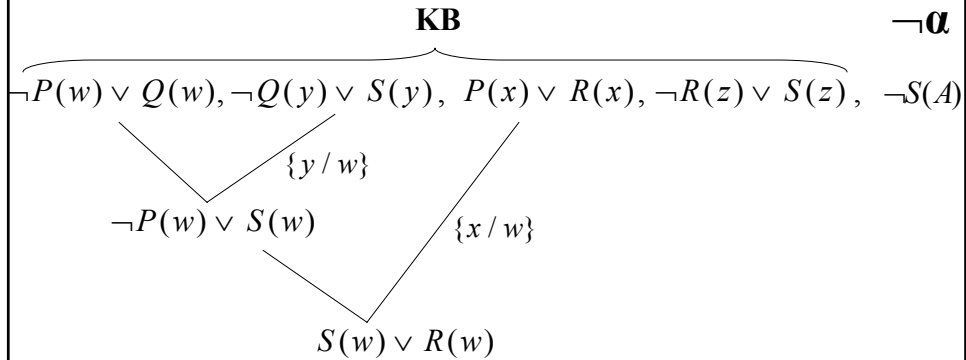
KB

$\neg \alpha$

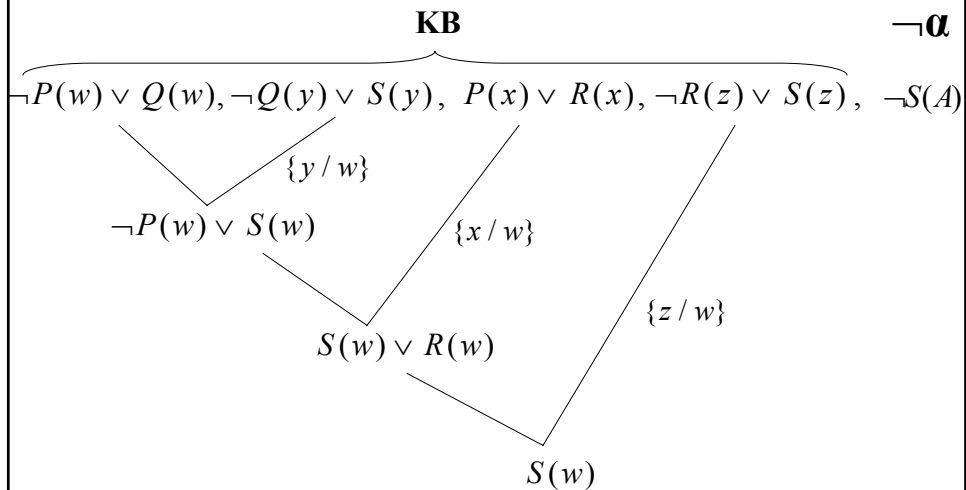
$\neg P(w) \vee Q(w), \neg Q(y) \vee S(y), P(x) \vee R(x), \neg R(z) \vee S(z), \neg S(A)$

$\neg P(w) \vee S(w)$
 $\{y / w\}$

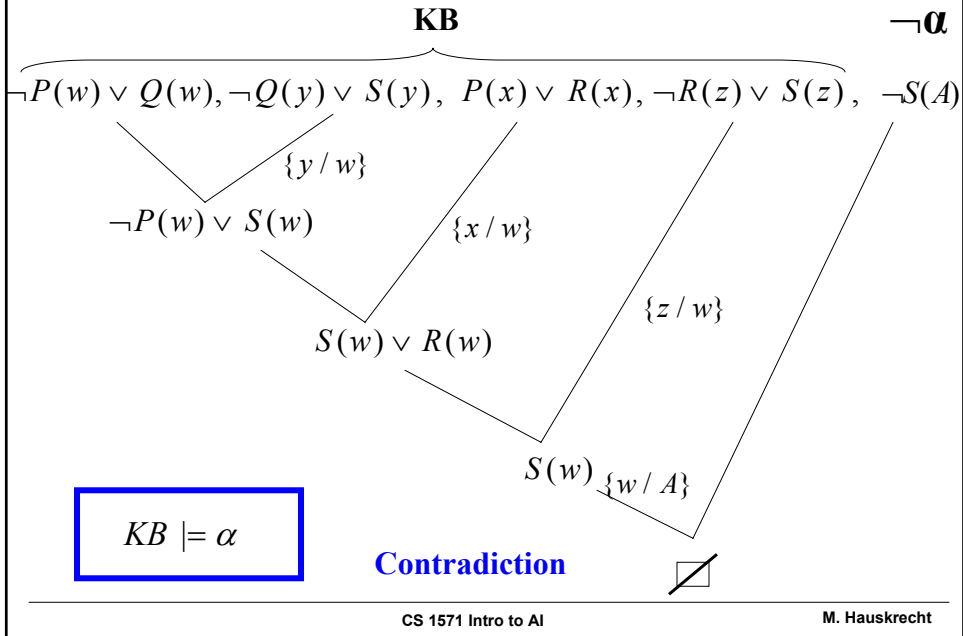
Resolution example



Resolution example



Resolution example



Dealing with equality

- Resolution works for first-order logic without equalities
- To incorporate equalities we need an additional inference rule
- **Demodulation rule**

$\sigma = \text{UNIFY}(\phi_i, t_1) \neq \text{fail}$

$$\frac{\phi_1 \vee \phi_2 \dots \vee \phi_k, \quad t_1 = t_2}{\text{SUBST}(\{\text{SUBST}(\sigma, t_1) / \text{SUBST}(\sigma, t_2)\}, \phi_1 \vee \dots \vee \phi_{i-1} \vee \phi_{i+1} \dots \vee \phi_k)}$$

- **Example:** $\frac{P(f(a)), f(x) = x}{P(a)}$
- **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

$$(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$$

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

- 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:

σ = a substitution s.t. $\forall i \text{ } SUBST(\sigma, \phi_i') = SUBST(\sigma, \phi_i)$

$$\frac{\phi_1', \phi_2', \dots, \phi_n', \quad \phi_1 \wedge \phi_2 \wedge \dots \phi_n \Rightarrow \tau}{SUBST(\sigma, \tau)}$$

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 α 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:

KB: R1: $Steamboat(x) \wedge Sailboat(y) \Rightarrow Faster(x, y)$

R2: $Sailboat(y) \wedge RowBoat(z) \Rightarrow Faster(y, z)$

R3: $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$

F1: $Steamboat(Titanic)$

F2: $Sailboat(Mistral)$

F3: $RowBoat(PondArrow)$

Theorem: $Faster(Titanic, PondArrow)$

?

Forward chaining example

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

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

?

Forward chaining example

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

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

Rule R1 is satisfied:

F4: $\text{Faster}(\text{Titanic}, \text{Mistral})$



Forward chaining example

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

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: $\text{Faster}(\text{Mistral}, \text{PondArrow})$ ←

Forward chaining example

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

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: $Steamboat(x) \wedge Sailboat(y) \Rightarrow Faster(x, y)$

R2: $Sailboat(y) \wedge RowBoat(z) \Rightarrow Faster(y, z)$

R3: $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$

F1: $Steamboat(Titanic)$

F2: $Sailboat(Mistral)$

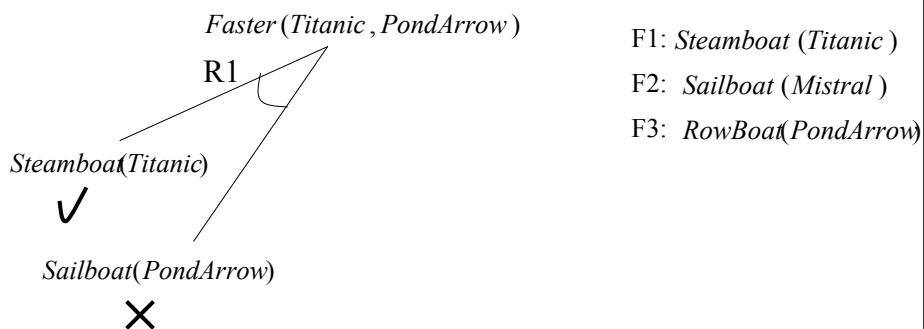
F3: $RowBoat(PondArrow)$

Theorem: $Faster(Titanic, PondArrow)$

CS 1571 Intro to AI

M. Hauskrecht

Backward chaining example



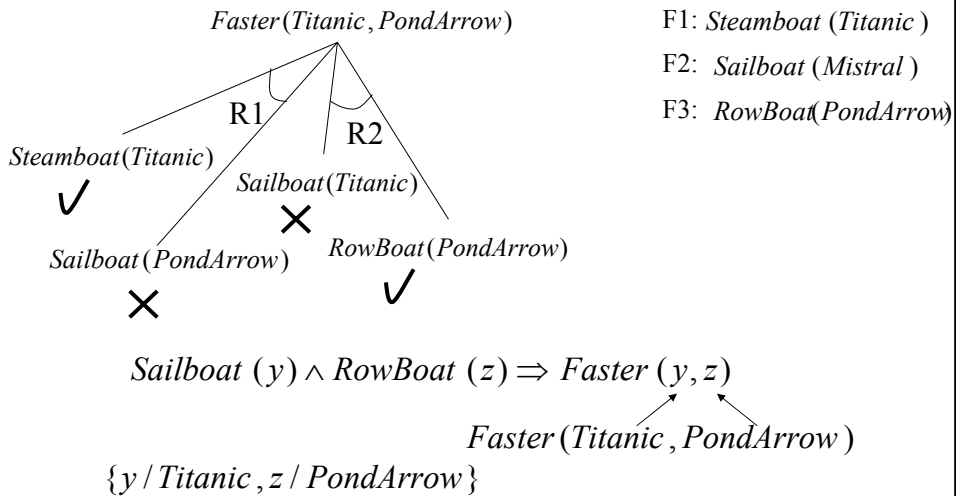
$Steamboat(x) \wedge Sailboat(y) \Rightarrow Faster(x, y)$

$Faster(Titanic, PondArrow)$
 $\{x / Titanic, y / PondArrow\}$

CS 1571 Intro to AI

M. Hauskrecht

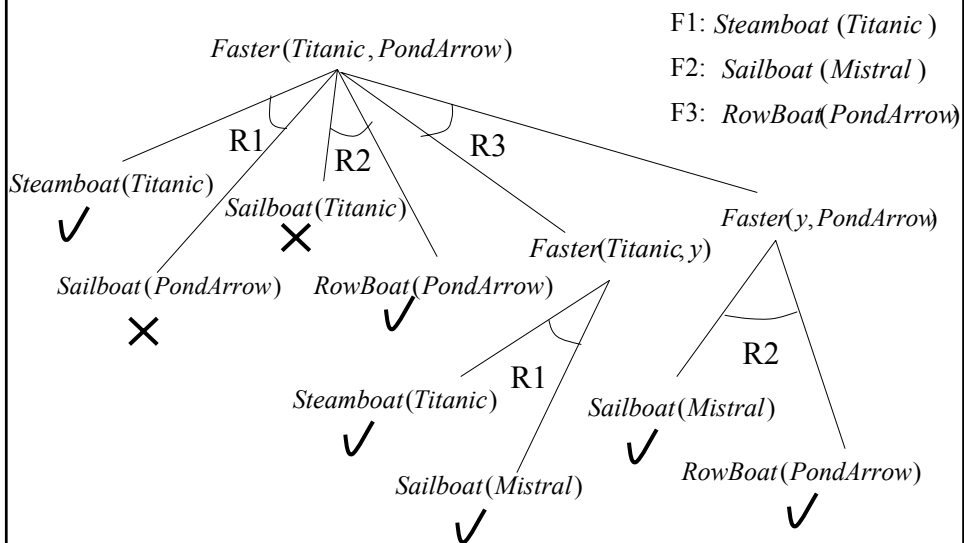
Backward chaining example



CS 1571 Intro to AI

M. Hauskrecht

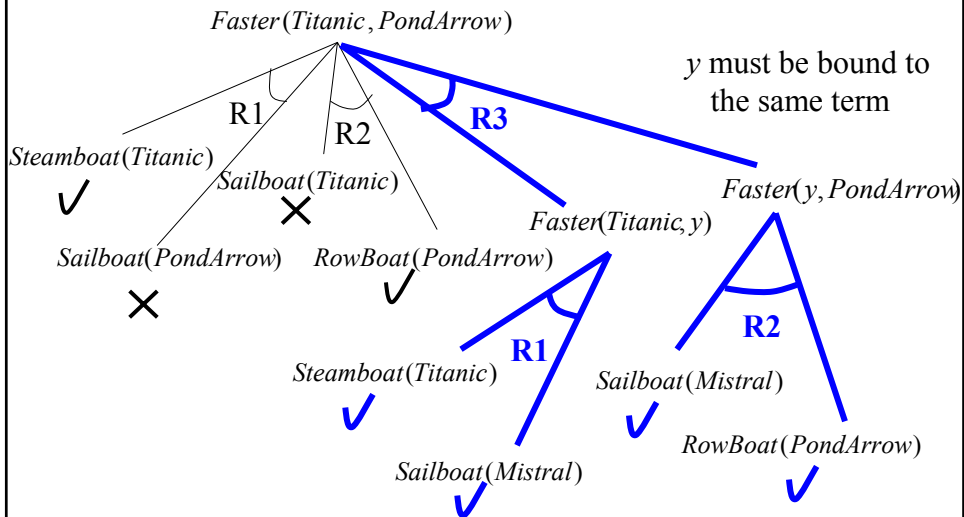
Backward chaining example



CS 1571 Intro to AI

M. Hauskrecht

Backward chaining



Backward chaining

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

