First-Order Logic

Dr. Steven Bethard

Computer and Information Sciences
University of Alabama at Birmingham

25 Feb 2016

Outline

1. First Order Logic
   - Core Components
   - Quantifiers
   - Example Translations
2. First-Order Logic Inference
   - Propositionalization
   - Generalized Modus Ponens
   - Forward Chaining
   - Backward Chaining
   - Prolog
   - Resolution

Outline

What's Wrong with Propositional Logic?

Translate:

All squares adjacent to pits are breezy

Problem: Propositional Logic

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
$B_{1,2} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{1,3})$
$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{2,1})$
$\ldots$

Solution: First Order Logic (Preview)

$\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \land Pit(r)$

What's Wrong with Propositional Logic?

Translate:

All squares adjacent to pits are breezy

Problem: Propositional Logic

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{2,1})$
$\ldots$

Solution: First Order Logic (Preview)

$\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$

What's Wrong with Propositional Logic?

Translate:

All squares adjacent to pits are breezy

Problem: Propositional Logic

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{2,1})$
$\ldots$

Solution: First Order Logic (Preview)

$\forall s \; \mathit{Breezy}(s) \Leftrightarrow \exists r \; \mathit{Adjacent}(r, s) \wedge \mathit{Pit}(r)$

Core Ideas of First Order Logic

Propositional Logic

- World consists of *facts*
- All facts are either true or false

First Order Logic

- World consists of *objects* and *relations*
- Statements that a relation R holds between objects X_1, $\ldots$, X_n are either true or false

Objects	people, numbers, houses, colors, years. . .
Relations	blonde, round, prime, multi-storied. . .
	brother-of, comes-between, has-color, after. . .

Core Ideas of First Order Logic

Propositional Logic

- World consists of *facts*
- All facts are either true or false

First Order Logic

- World consists of *objects* and *relations*
- Statements that a relation R holds between objects $X_1, \ldots, X_n$ are either true or false

Objects	people, numbers, houses, colors, years. . .
Relations	blonde, round, prime, multi-storied. . .
	brother-of, comes-between, has-color, after. . .

Core Ideas of First Order Logic

Propositional Logic

- World consists of *facts*
- All facts are either true or false

First Order Logic

- World consists of *objects* and *relations*
- Statements that a relation R holds between objects X_1, \ldots, X_n are either true or false

Objects	people, numbers, houses, colors, years. . .
Relations	blonde, round, prime, multi-storied. . . brother-of, comes-between, has-color, after. . .

Constants

Key Idea

Constants represent named objects in the world

Examples

RichardTheLionheart
RonaldMcDonald
Blue
42
12pm

Functions

Key Idea

Functions relate object(s) to *exactly one* other object

Examples

$LeftLegOf(x)$
$LengthOf(x)$
$SquareRoot(x)$, i.e. $\sqrt{x}$
$Sum(x, y)$, i.e. $x + y$
$Intersection(x, y)$, i.e. $x \cap y$

Predicates

Key Idea

Predicates describe relations between objects (or a property of a single object)

Examples

$Person(x)$

$Female(x)$

$BrotherOf(x, y)$

$Positive(x)$, i.e. $x > 0$

$MemberOf(x, y)$, i.e. $x \in y$

$SubsetOf(x, y)$, i.e. $x \subseteq y$

Connectives

Connectives in First-Order Logic

- Used to construct more complex sentences
- Semantics match those of Propositional Logic

Examples

$\neg Brother(LeftLeg(Richard), John)$

$Positive(42) \land LessThan(42, 100)$

$Male(Pat) \lor Female(Pat)$

$\neg LaysEggs(Whale) \Rightarrow \neg Bird(Whale)$

Truth in First Order Logic

Key Ideas

- Relations are (possibly infinite) sets of tuples
- $R(Term_1, \ldots, Term_n)$ true iff $\langle Term_1, \ldots, Term_n \rangle \in R$

Example

Given relation $SquareRoot = \{\langle 1, 1\rangle, \langle 4, 2\rangle, \langle 9, 3\rangle, \ldots\}$

$SquareRoot(4, 2)$? true

$SquareRoot(2, 2)$? false

Truth in First Order Logic

Key Ideas

- Relations are (possibly infinite) sets of tuples
- $R(Term_1, \ldots, Term_n)$ true iff $\langle Term_1, \ldots, Term_n \rangle \in R$

Example

Given relation $SquareRoot = \{\langle 1, 1 \rangle, \langle 4, 2 \rangle, \langle 9, 3 \rangle, \ldots\}$

$SquareRoot(4, 2)$? true

$SquareRoot(2, 2)$? false

Truth in First Order Logic

Key Ideas

- Relations are (possibly infinite) sets of tuples
- $R(Term_1, \ldots, Term_n)$ true iff $\langle Term_1, \ldots, Term_n \rangle \in R$

Example

Given relation $SquareRoot = \{\langle 1, 1\rangle, \langle 4, 2\rangle, \langle 9, 3\rangle, \ldots\}$

$SquareRoot(4, 2)$? true

$SquareRoot(2, 2)$? false

Truth in First Order Logic

Key Ideas

- Relations are (possibly infinite) sets of tuples
- $R(Term_1, \ldots, Term_n)$ true iff $\langle Term_1, \ldots, Term_n \rangle \in R$

Example

Given relation $SquareRoot = \{\langle 1, 1\rangle, \langle 4, 2\rangle, \langle 9, 3\rangle, \ldots\}$

$SquareRoot(4, 2)$? true

$SquareRoot(2, 2)$? false

Truth in First Order Logic

Key Ideas

- Relations are (possibly infinite) sets of tuples
- $R(Term_1, \ldots, Term_n)$ true iff $\langle Term_1, \ldots, Term_n \rangle \in R$

Example

Given relation $SquareRoot = \{\langle 1, 1\rangle, \langle 4, 2\rangle, \langle 9, 3\rangle, \ldots\}$

$SquareRoot(4, 2)$? true

$SquareRoot(2, 2)$? false

Quantifiers

Predicating over Constants

If I know:

$$\neg LaysEggs(Whale) \Rightarrow \neg Bird(Whale)$$

What can I say here?

$$\neg LaysEggs(Steve) \Rightarrow ?$$

Nothing!

Quantifiers and Variables

Quantifiers allow statements about classes of objects, e.g.

$$\forall x\ \neg LaysEggs(x) \Rightarrow \neg Bird(x)$$

Quantifiers

Predicating over Constants

If I know:

$$\neg LaysEggs(Whale) \Rightarrow \neg Bird(Whale)$$

What can I say here?

$$\neg LaysEggs(Steve) \Rightarrow ?$$

Nothing!

Quantifiers and Variables

Quantifiers allow statements about classes of objects, e.g.

$$\forall x \; \neg LaysEggs(x) \Rightarrow \neg Bird(x)$$

Quantifiers

Predicating over Constants

If I know:

$$\neg LaysEggs(Whale) \Rightarrow \neg Bird(Whale)$$

What can I say here?

$$\neg LaysEggs(Steve) \Rightarrow ?$$

Nothing!

Quantifiers and Variables

Quantifiers allow statements about classes of objects, e.g.

$$\forall x \; \neg LaysEggs(x) \Rightarrow \neg Bird(x)$$

Universal Quantification ($\forall$)

Definition

$\forall x\ P$ is true in model m iff:

P is true when we bind x to each of the objects in m

Example

So $\forall x\ Bird(x) \Rightarrow LaysEggs(x)$ is true because:

$Bird(Swallow) \Rightarrow LaysEggs(Swallow)$ is true

$Bird(Emu) \Rightarrow LaysEggs(Emu)$ is true

$Bird(Badger) \Rightarrow LaysEggs(Badger)$ is true

...

But $\forall x\ LaysEggs(x) \Rightarrow Bird(x)$ is false because:

$LaysEggs(Platypus) \Rightarrow Bird(Platypus)$ is false

Universal Quantification ($\forall$)

Definition

$\forall x\ P$ is true in model m iff:

P is true when we bind x to each of the objects in m

Example

So $\forall x\ Bird(x) \Rightarrow LaysEggs(x)$ is true because:

$Bird(Swallow) \Rightarrow LaysEggs(Swallow)$ is true

$Bird(Emu) \Rightarrow LaysEggs(Emu)$ is true

$Bird(Badger) \Rightarrow LaysEggs(Badger)$ is true

$\ldots$

But $\forall x\ LaysEggs(x) \Rightarrow Bird(x)$ is false because:

$LaysEggs(Platypus) \Rightarrow Bird(Platypus)$ is false

Universal Quantification ($\forall$)

Definition

$\forall x\ P$ is true in model m iff:

P is true when we bind x to each of the objects in m

Example

So $\forall x\ \mathit{Bird}(x) \Rightarrow \mathit{LaysEggs}(x)$ is true because:

$\mathit{Bird}(\mathit{Swallow}) \Rightarrow \mathit{LaysEggs}(\mathit{Swallow})$ is true

$\mathit{Bird}(\mathit{Emu}) \Rightarrow \mathit{LaysEggs}(\mathit{Emu})$ is true

$\mathit{Bird}(\mathit{Badger}) \Rightarrow \mathit{LaysEggs}(\mathit{Badger})$ is true

$\ldots$

But $\forall x\ \mathit{LaysEggs}(x) \Rightarrow \mathit{Bird}(x)$ is false because:

$\mathit{LaysEggs}(\mathit{Platypus}) \Rightarrow \mathit{Bird}(\mathit{Platypus})$ is false

Universal Quantification ($\forall$)

Definition

$\forall x\ P$ is true in model m iff:

P is true when we bind x to each of the objects in m

Example

So $\forall x\ Bird(x) \Rightarrow LaysEggs(x)$ is true because:

$Bird(Swallow) \Rightarrow LaysEggs(Swallow)$ is true

$Bird(Emu) \Rightarrow LaysEggs(Emu)$ is true

$Bird(Badger) \Rightarrow LaysEggs(Badger)$ is true

$\ldots$

But $\forall x\ LaysEggs(x) \Rightarrow Bird(x)$ is false because:

$LaysEggs(Platypus) \Rightarrow Bird(Platypus)$ is false

Universal Quantification in Translation

Common Mistake

What does this mean?

$\forall x\ Bird(x) \wedge LaysEggs(x)$

Answer:

Everything is a bird and everything lays eggs

Intended statement:

$\forall x\ Bird(x) \Rightarrow LaysEggs(x)$

Rule of Thumb

Use implication ($\Rightarrow$) with universal quantification ($\forall$)

Universal Quantification in Translation

Common Mistake

What does this mean?

$\forall x \; Bird(x) \wedge LaysEggs(x)$

Answer:

Everything is a bird and everything lays eggs

Intended statement:

$\forall x \; Bird(x) \Rightarrow LaysEggs(x)$

Rule of Thumb

Use implication ($\Rightarrow$) with universal quantification ($\forall$)

Universal Quantification in Translation

Common Mistake

What does this mean?

$\forall x \, Bird(x) \wedge LaysEggs(x)$

Answer:

Everything is a bird and everything lays eggs

Intended statement:

$\forall x \, Bird(x) \Rightarrow LaysEggs(x)$

Rule of Thumb

Use implication ($\Rightarrow$) with universal quantification ($\forall$)

Existential Quantification

Definition

$\exists x\ P$ is true in model m iff:

P is true when we bind x to any of the objects in m

Example

So $\exists x\ Mammal(x) \land LaysEggs(x)$ is true because:

$Mammal(Platypus) \land LaysEggs(Platypus)$ is true

But $\exists x\ Mammal(x) \land \neg Animal(x)$ is false because:

$Mammal(Person) \land \neg Animal(Person)$ is false

$Mammal(Platypus) \land \neg Animal(Platypus)$ is false

$Mammal(Spam) \land \neg Animal(Spam)$ is false

...

Existential Quantification

Definition

$\exists x\ P$ is true in model m iff:

P is true when we bind x to any of the objects in m

Example

So $\exists x\ Mammal(x) \land LaysEggs(x)$ is true because:

$Mammal(Platypus) \land LaysEggs(Platypus)$ is true

But $\exists x\ Mammal(x) \land \neg Animal(x)$ is false because:

$Mammal(Person) \land \neg Animal(Person)$ is false

$Mammal(Platypus) \land \neg Animal(Platypus)$ is false

$Mammal(Spam) \land \neg Animal(Spam)$ is false

...

Existential Quantification

Definition

$\exists x\ P$ is true in model m iff:

P is true when we bind x to any of the objects in m

Example

So $\exists x\ Mammal(x) \wedge LaysEggs(x)$ is true because:

$Mammal(Platypus) \wedge LaysEggs(Platypus)$ is true

But $\exists x\ Mammal(x) \wedge \neg Animal(x)$ is false because:

$Mammal(Person) \wedge \neg Animal(Person)$ is false

$Mammal(Platypus) \wedge \neg Animal(Platypus)$ is false

$Mammal(Spam) \wedge \neg Animal(Spam)$ is false

...

Existential Quantification

Definition

$\exists x\ P$ is true in model m iff:

P is true when we bind x to any of the objects in m

Example

So $\exists x\ Mammal(x) \wedge LaysEggs(x)$ is true because:

$Mammal(Platypus) \wedge LaysEggs(Platypus)$ is true

But $\exists x\ Mammal(x) \wedge \neg Animal(x)$ is false because:

$Mammal(Person) \wedge \neg Animal(Person)$ is false

$Mammal(Platypus) \wedge \neg Animal(Platypus)$ is false

$Mammal(Spam) \wedge \neg Animal(Spam)$ is false

...

Existential Quantification

Definition

$\exists x\ P$ is true in model m iff:

P is true when we bind x to any of the objects in m

Example

So $\exists x\ Mammal(x) \land LaysEggs(x)$ is true because:

$Mammal(Platypus) \land LaysEggs(Platypus)$ is true

But $\exists x\ Mammal(x) \land \neg Animal(x)$ is false because:

$Mammal(Person) \land \neg Animal(Person)$ is false

$Mammal(Platypus) \land \neg Animal(Platypus)$ is false

$Mammal(Spam) \land \neg Animal(Spam)$ is false

...

Existential Quantification in Translation

Common Mistake

What does this mean?

$\exists x \; Mammal(x) \Rightarrow LaysEggs(x)$

Answer:

There is something that is not a mammal or lays eggs

Intended statement:

$\exists x \; Mammal(x) \land LaysEggs(x)$

Rule of Thumb

Use conjunction ($\land$) with existential quantification ($\exists$)

Existential Quantification in Translation

Common Mistake

What does this mean?

$\exists x\ Mammal(x) \Rightarrow LaysEggs(x)$

Answer:

There is something that is not a mammal or lays eggs

Intended statement:

$\exists x\ Mammal(x) \land LaysEggs(x)$

Rule of Thumb

Use conjunction ($\land$) with existential quantification ($\exists$)

Existential Quantification in Translation

Common Mistake

What does this mean?

$\exists x \; Mammal(x) \Rightarrow LaysEggs(x)$

Answer:

There is something that is not a mammal or lays eggs

Intended statement:

$\exists x \; Mammal(x) \wedge LaysEggs(x)$

Rule of Thumb

Use conjunction ($\wedge$) with existential quantification ($\exists$)

Quantifier Properties

Nesting Quantifiers

Mixed quantifiers cannot be exchanged:

- $\forall x \; \exists y \; Loves(x, y)$ "everyone loves someone"
- $\exists y \; \forall x \; Loves(x, y)$ "one person is loved by everyone"

Relation between $\forall$ and $\exists$

Conversion is roughly like DeMorgan's:

- $\forall x \; Enjoys(x, AI) \equiv \neg\exists x \; \neg Enjoys(x, AI)$
- $\exists x \; Enjoys(x, DB) \equiv \neg\forall x \; \neg Enjoys(x, DB)$

Quantifier Properties

Nesting Quantifiers

Mixed quantifiers cannot be exchanged:

- $\forall x\ \exists y\ Loves(x, y)$ "everyone loves someone"
- $\exists y\ \forall x\ Loves(x, y)$ "one person is loved by everyone"

Relation between ∀ and ∃

Conversion is roughly like DeMorgan's:

- $\forall x\ Enjoys(x, AI) \equiv \neg\exists x\ \neg Enjoys(x, AI)$
- $\exists x\ Enjoys(x, DB) \equiv \neg\forall x\ \neg Enjoys(x, DB)$

Equality Relations

Problem

What's wrong with this definition?

$$\forall x, y \; Sibling(x, y) \Leftrightarrow \exists p \; Parent(p, x) \wedge Parent(p, y)$$

Both x and y can be the same thing!

$$Sibling(Steve, Steve)$$

Solution: Equality

Specify when two variables refer to the same objects:

$$\forall x, y \; Sibling(x, y) \Leftrightarrow x \neq y \wedge \exists p \; Parent(p, x) \wedge Parent(p, y)$$

Equality Relations

Problem

What's wrong with this definition?

$$\forall x, y \; Sibling(x, y) \Leftrightarrow \exists p \; Parent(p, x) \wedge Parent(p, y)$$

Both x and y can be the same thing!

$$Sibling(Steve, Steve)$$

Solution: Equality

Specify when two variables refer to the same objects:

$$\forall x, y \; Sibling(x, y) \Leftrightarrow x \neq y \wedge \exists p \; Parent(p, x) \wedge Parent(p, y)$$

Equality Relations

Problem

What's wrong with this definition?

$\forall x, y \; Sibling(x, y) \Leftrightarrow \exists p \; Parent(p, x) \land Parent(p, y)$

Both x and y can be the same thing!

$Sibling(Steve, Steve)$

Solution: Equality

Specify when two variables refer to the same objects:

$\forall x, y \; Sibling(x, y) \Leftrightarrow x \neq y \land \exists p \; Parent(p, x) \land Parent(p, y)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \wedge LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \wedge \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \wedge Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \wedge Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \wedge Swim(x) \wedge \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x \; Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x \; Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x \; Mammal(x) \wedge LaysEggs(x)$
- Some birds don't fly
  $\exists x \; Bird(x) \wedge \neg Flies(x)$
- Animals that fly have wings
  $\forall x \; Animal(x) \wedge Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x \; Animal(x) \wedge Swim(x) \Rightarrow HasFins(x)$
  $\exists x \; Animal(x) \wedge Swim(x) \wedge \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \wedge LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \wedge \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \wedge Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \wedge Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \wedge Swim(x) \wedge \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \land LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \land \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \land Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \land Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \land Swim(x) \land \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \wedge LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \wedge \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \wedge Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \wedge Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \wedge Swim(x) \wedge \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \wedge LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \wedge \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \wedge Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \wedge Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \wedge Swim(x) \wedge \neg HasFins(x)$

Animal Kingdom Translations

- All horses are mammals
  $\forall x\ Horse(x) \Rightarrow Mammal(x)$
- All birds have wings
  $\forall x\ Bird(x) \Rightarrow HasWings(x)$
- Some mammals lay eggs
  $\exists x\ Mammal(x) \land LaysEggs(x)$
- Some birds don't fly
  $\exists x\ Bird(x) \land \neg Flies(x)$
- Animals that fly have wings
  $\forall x\ Animal(x) \land Flies(x) \Rightarrow HasWings(x)$
- Not all swimming animals have fins
  $\neg\forall x\ Animal(x) \land Swim(x) \Rightarrow HasFins(x)$
  $\exists x\ Animal(x) \land Swim(x) \land \neg HasFins(x)$

Animal Kingdom Translations

- An animal gives birth to animals of the same species

  $$\forall x, y \; Animal(x) \wedge GivesBirth(x, y) \Rightarrow$$
  $$Animal(y) \wedge Species(x) = Species(y) \wedge x \neq y$$

- Bats have exactly two wings

  $$\forall x \; Bat(x) \Rightarrow$$
  $$\exists y, z \; HasWing(x, y) \wedge HasWing(x, z) \wedge y \neq z \wedge$$
  $$\forall w \; HasWing(x, w) \Rightarrow w = y \vee w = z$$

Animal Kingdom Translations

- An animal gives birth to animals of the same species
  $$\forall x, y\ Animal(x) \land GivesBirth(x, y) \Rightarrow$$
  $$Animal(y) \land Species(x) = Species(y) \land x \neq y$$

- Bats have exactly two wings
  $$\forall x\ Bat(x) \Rightarrow$$
  $$\exists y, z\ HasWing(x, y) \land HasWing(x, z) \land y \neq z \land$$
  $$\forall w\ HasWing(x, w) \Rightarrow w = y \lor w = z$$

Animal Kingdom Translations

- An animal gives birth to animals of the same species
  $\forall x, y\ Animal(x) \land GivesBirth(x, y) \Rightarrow$
  $\quad Animal(y) \land Species(x) = Species(y) \land x \neq y$

- Bats have exactly two wings
  $\forall x\ Bat(x) \Rightarrow$
  $\quad \exists y, z\ HasWing(x, y) \land HasWing(x, z) \land y \neq z \land$
  $\quad\quad \forall w\ HasWing(x, w) \Rightarrow w = y \lor w = z$

Wumpus World Translations

- There is a breeze in $[3,1]$
  $Breezy([3,1])$

- The Wumpus is lives in $[2,2]$
  $Home(Wumpus) = [2,2]$

- If you are in the Wumpus's square, he eats you
  $\forall t\ Location(Agent, Home(Wumpus), t) \Rightarrow HasEaten(Wumpus, Agent, t)$

- You should grab the gold when you are in its square
  $\forall s, t\ HasGold(s) \land Location(Agent, s, t) \Rightarrow BestAction(Grab, Agent, t)$

Wumpus World Translations

- There is a breeze in $[3,1]$
  $Breezy([3,1])$

- The Wumpus is lives in $[2,2]$
  $Home(Wumpus) = [2,2]$

- If you are in the Wumpus's square, he eats you
  $\forall t\ Location(Agent, Home(Wumpus), t) \Rightarrow$
  $HasEaten(Wumpus, Agent, t)$

- You should grab the gold when you are in its square
  $\forall s, t\ HasGold(s) \wedge Location(Agent, s, t) \Rightarrow$
  $BestAction(Grab, Agent, t)$

Wumpus World Translations

- There is a breeze in $[3,1]$
  $Breezy([3,1])$

- The Wumpus is lives in $[2,2]$
  $Home(Wumpus) = [2,2]$

- If you are in the Wumpus's square, he eats you
  $\forall t\ Location(Agent, Home(Wumpus), t) \Rightarrow$
  $HasEaten(Wumpus, Agent, t)$

- You should grab the gold when you are in its square
  $\forall s, t\ HasGold(s) \wedge Location(Agent, s, t) \Rightarrow$
  $BestAction(Grab, Agent, t)$

Wumpus World Translations

- There is a breeze in $[3,1]$
  $Breezy([3,1])$

- The Wumpus is lives in $[2,2]$
  $Home(Wumpus) = [2,2]$

- If you are in the Wumpus's square, he eats you
  $\forall t\ Location(Agent, Home(Wumpus), t) \Rightarrow$
  $HasEaten(Wumpus, Agent, t)$

- You should grab the gold when you are in its square
  $\forall s, t\ HasGold(s) \wedge Location(Agent, s, t) \Rightarrow$
  $BestAction(Grab, Agent, t)$

Wumpus World Translations

- There is a breeze in $[3,1]$
  $Breezy([3,1])$

- The Wumpus is lives in $[2,2]$
  $Home(Wumpus) = [2,2]$

- If you are in the Wumpus's square, he eats you
  $\forall t\ Location(Agent, Home(Wumpus), t) \Rightarrow HasEaten(Wumpus, Agent, t)$

- You should grab the gold when you are in its square
  $\forall s, t\ HasGold(s) \land Location(Agent, s, t) \Rightarrow BestAction(Grab, Agent, t)$

Wumpus World Translations

Diagnostic Rules

From effect, determine cause:

$\forall y\ Breezy(y) \Rightarrow (\exists x\ Pit(x) \land Adjacent(x, y))$

Causal Rules

From cause, determine effect:

$\forall x, y\ (Pit(x) \land Adjacent(x, y)) \Rightarrow Breezy(y)$

Definition Rules

Bidirectional:

$\forall y\ Breezy(y) \Leftrightarrow (\exists x\ Pit(x) \land Adjacent(x, y))$

Wumpus World Translations

Diagnostic Rules

From effect, determine cause:

$$\forall y\ Breezy(y) \Rightarrow (\exists x\ Pit(x) \wedge Adjacent(x, y))$$

Causal Rules

From cause, determine effect:

$$\forall x, y\ (Pit(x) \wedge Adjacent(x, y)) \Rightarrow Breezy(y)$$

Definition Rules

Bidirectional:

$$\forall y\ Breezy(y) \Leftrightarrow (\exists x\ Pit(x) \wedge Adjacent(x, y))$$

Wumpus World Translations

Diagnostic Rules

From effect, determine cause:

$$\forall y \; Breezy(y) \Rightarrow (\exists x \; Pit(x) \land Adjacent(x, y))$$

Causal Rules

From cause, determine effect:

$$\forall x, y \; (Pit(x) \land Adjacent(x, y)) \Rightarrow Breezy(y)$$

Definition Rules

Bidirectional:

$$\forall y \; Breezy(y) \Leftrightarrow (\exists x \; Pit(x) \land Adjacent(x, y))$$

Harry Potter’s 7 Potions Puzzle

Danger lies before you, while safety lies behind,
Two of us will help you, whichever you would find.
One among us seven will let you move ahead,
Another will transport the drinker back instead.
Two among our number hold only nettle wine,
Three of us are killers, waiting hidden in line.
Choose, unless you wish to stay here forevermore,
To help you in your choice, we give you these clues four:
First, however slyly the poison tries to hide
You will always find some on nettle wine’s left side;
Second, different are those who stand at either end,
But if you would move forward, niether is your friend;
Third, as you see clearly, all are different size,
Neither dwarf nor giant holds death in their insides;
Fourth, the second left and second on the right
Are twins once you taste them, though different at first sight.

One Harry Potter's 7 Potions Solution

$$
\begin{aligned}
&\forall p\ \mathit{Potions}(p) \Leftrightarrow \\
&\quad \mathit{Permutation}(p, [\mathit{Forward}, \mathit{Backward}, \mathit{Wine}, \mathit{Wine}, \\
&\qquad\qquad\qquad\qquad \mathit{Poison}, \mathit{Poison}, \mathit{Poison}]) \\
&\quad \mathit{PoisonIsLeftOfWine}(p) \wedge \\
&\quad \mathit{EndsAreDifferent}(p) \wedge \mathit{EndsAreNotForward}(p) \wedge \\
&\quad \mathit{SmallestIsNotPoison}(p) \wedge \mathit{LargestIsNotPoison}(p) \\
&\quad \mathit{SecondsAreTheSame}(p) \wedge \\
&\ldots
\end{aligned}
$$

One Harry Potter's 7 Potions Solution

$$
\begin{aligned}
&\forall p\ \mathit{Potions}(p) \Leftrightarrow\\
&\quad \mathit{Permutation}(p, [\mathit{Forward}, \mathit{Backward}, \mathit{Wine}, \mathit{Wine},\\
&\qquad\qquad\qquad\qquad \mathit{Poison}, \mathit{Poison}, \mathit{Poison}])\\
&\quad \mathit{PoisonIsLeftOfWine}(p) \wedge\\
&\quad \mathit{EndsAreDifferent}(p) \wedge \mathit{EndsAreNotForward}(p) \wedge\\
&\quad \mathit{SmallestIsNotPoison}(p) \wedge \mathit{LargestIsNotPoison}(p)\\
&\quad \mathit{SecondsAreTheSame}(p) \wedge
\end{aligned}
$$

$\ldots$

$$
\begin{aligned}
&\forall p\ \mathit{EndsAreDifferent}(p) \Leftrightarrow\\
&\quad \exists p_1, p_2, \ldots, p_7\ p = [p_1, p_2, p_3, p_4, p_5, p_6, p_7] \wedge p_1 \neq p_7
\end{aligned}
$$

$\ldots$

One Harry Potter's 7 Potions Solution

$$\forall p\ \mathit{Potions}(p) \Leftrightarrow$$
$$\mathit{Permutation}(p, [\mathit{Forward}, \mathit{Backward}, \mathit{Wine}, \mathit{Wine}, \mathit{Poison}, \mathit{Poison}, \mathit{Poison}])$$
$$\mathit{PoisonIsLeftOfWine}(p) \wedge$$
$$\mathit{EndsAreDifferent}(p) \wedge \mathit{EndsAreNotForward}(p) \wedge$$
$$\mathit{SmallestIsNotPoison}(p) \wedge \mathit{LargestIsNotPoison}(p)$$
$$\mathit{SecondsAreTheSame}(p) \wedge$$

...

$$\forall p\ \mathit{EndsAreDifferent}(p) \Leftrightarrow$$
$$\exists p_1, p_2, \ldots, p_7\ p = [p_1, p_2, p_3, p_4, p_5, p_6, p_7] \wedge p_1 \neq p_7$$

...

$$\forall p\ \mathit{LargestIsNotPoison}(p) \Leftrightarrow$$
$$\exists p_i\ \mathit{Largest}(p, p_i) \wedge p_i \neq \mathit{Poison}$$

Prolog Demo

Outline

1. First Order Logic
   - Core Components
   - Quantifiers
   - Example Translations
2. First-Order Logic Inference
   - Propositionalization
   - Generalized Modus Ponens
   - Forward Chaining
   - Backward Chaining
   - Prolog
   - Resolution

Universal Instantiation

Key Idea

If we know $\forall x\ P(x)$, then we can conclude:

- $P(\mathit{Badger})$
- $P(\mathit{Spam})$
- $\ldots$

Formal Rule

Given a ground term g:

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

Universal Instantiation

Key Idea

If we know $\forall x\ P(x)$, then we can conclude:

- $P(Badger)$
- $P(Spam)$
- $\ldots$

Formal Rule

Given a ground term g:

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

Existential Instantiation

Key Idea

If we know $\exists x\, P(x)$, then we can just give a name to x:

- $P(\mathit{ThingThatPIsTrueFor})$

This works as long as the name isn't already in use

Formal Rule

Given a constant k that is not in the knowledge base:

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

The constant k is called a Skolem constant

Existential Instantiation

Key Idea

If we know $\exists x\, P(x)$, then we can just give a name to x:

- $P(\mathit{ThingThatPIsTrueFor})$

This works as long as the name isn't already in use

Formal Rule

Given a constant k that is not in the knowledge base:

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

The constant k is called a Skolem constant

Reduction to Propositional Logic

First-Order Logic

$\exists x \; Mammal(x) \land LaysEggs(x)$
$\forall x \; Mammal(x) \Rightarrow WarmBlooded(x)$
$Mammal(Platypus)$
$\neg WarmBlooded(Crocodile)$

Propositional Logic

$Mammal9X23B \land LaysEggs9X23B$
$MammalPlatypus \Rightarrow WarmBloodedPlatypus$
$MammalCrocodile \Rightarrow WarmBloodedCrocodile$
$MammalPlatypus$
$\neg WarmBloodedCrocodile$

Reduction to Propositional Logic

First-Order Logic

$\exists x \; Mammal(x) \wedge LaysEggs(x)$
$\forall x \; Mammal(x) \Rightarrow WarmBlooded(x)$
$Mammal(Platypus)$
$\neg WarmBlooded(Crocodile)$

Propositional Logic

$Mammal9X23B \wedge LaysEggs9X23B$
$MammalPlatypus \Rightarrow WarmBloodedPlatypus$
$MammalCrocodile \Rightarrow WarmBloodedCrocodile$
$MammalPlatypus$
$\neg WarmBloodedCrocodile$

Simple First-Order Inference

Simple Approach

- Remove ∀ and ∃
- Treat all first-order terms as simple symbols
- Solve using resolution for propositional logic

Example: $WarmBlooded(Platypus)$?

$$\frac{\neg WarmBlooded(Platypus) \qquad \neg Mammal(Platypus) \lor WarmBlooded(Platypus)}{\neg Mammal(Platypus)}$$

$$\frac{\neg Mammal(Platypus) \qquad Mammal(Platypus)}{false}$$

Simple First-Order Inference

Simple Approach

- Remove $\forall$ and $\exists$
- Treat all first-order terms as simple symbols
- Solve using resolution for propositional logic

Example: $WarmBlooded(Platypus)$?

$$\frac{\neg WarmBlooded(Platypus) \qquad \neg Mammal(Platypus) \vee WarmBlooded(Platypus)}{\neg Mammal(Platypus)}$$

$$\frac{\neg Mammal(Platypus) \qquad Mammal(Platypus)}{false}$$

Simple First-Order Inference

Problem: Infinite Terms

- $Mammal(Steve)$
- $Mammal(Mother(Steve))$
- $Mammal(Mother(Mother(Steve)))$
- $\ldots$

Solution: Iterative Deepening

- Try proof with terms up to depth 1
- Try proof with terms up to depth 2
- $\ldots$

Proof found if exists, else infinite loop (semidecidable)

Simple First-Order Inference

Problem: Infinite Terms

- $Mammal(Steve)$
- $Mammal(Mother(Steve))$
- $Mammal(Mother(Mother(Steve)))$
- ...

Solution: Iterative Deepening

- Try proof with terms up to depth 1
- Try proof with terms up to depth 2
- ...

Proof found if exists, else infinite loop (semidecidable)

Simple First-Order Inference

Problem: Infinite Terms

- $Mammal(Steve)$
- $Mammal(Mother(Steve))$
- $Mammal(Mother(Mother(Steve)))$
- ...

Solution: Iterative Deepening

- Try proof with terms up to depth 1
- Try proof with terms up to depth 2
- ...

Proof found if exists, else infinite loop (semidecidable)

Simple First-Order Inference

Problem: Infinite Terms

- $Mammal(Steve)$
- $Mammal(Mother(Steve))$
- $Mammal(Mother(Mother(Steve)))$
- $\ldots$

Solution: Iterative Deepening

- Try proof with terms up to depth 1
- Try proof with terms up to depth 2
- $\ldots$

Proof found if exists, else infinite loop (semidecidable)

Simple First-Order Inference

Problem: Infinite Terms

- $Mammal(Steve)$
- $Mammal(Mother(Steve))$
- $Mammal(Mother(Mother(Steve)))$
- $\ldots$

Solution: Iterative Deepening

- Try proof with terms up to depth 1
- Try proof with terms up to depth 2
- $\ldots$

Proof found if exists, else infinite loop (semidecidable)

Problems with Propositionalization

Prove: $WarmBlooded(Scooby)$

$Dog(Scooby)$
$Dog(Scrappy)$
$\forall x\ Dog(x) \Rightarrow Mammal(x)$
$\forall y\ Mammal(y) \Rightarrow WarmBlooded(y)$

Problem: Many Irrelevant Facts Produced

$Dog(Scrappy)$
$Dog(Scrappy) \Rightarrow Mammal(Scrappy)$
$Mammal(Scrappy) \Rightarrow WarmBlooded(Scrappy)$

Generalized Modus Ponens

Definition

$$
\frac{\begin{array}{l} p_1' \\ p_2' \\ \ldots \\ p_n' \\ p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q \\ \theta : \forall i \ \textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i) \end{array}}{\textsc{Subst}(\theta, q)}
$$

Example

$$
\frac{\begin{array}{l} Odd(17) \\ \forall x \ Odd(x) \Rightarrow Mod(x, 2, 1) \\ \theta = \{x = 17\} \end{array}}{Mod(17, 2, 1)}
$$

Generalized Modus Ponens

Definition

p'_1

p'_2

$\ldots$

p'_n

$p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$

$\theta : \forall i \; \textsc{Subst}(\theta, p'_i) = \textsc{Subst}(\theta, p_i)$

$\textsc{Subst}(\theta, q)$

Example

$Odd(17)$

$\forall x \; Odd(x) \Rightarrow Mod(x, 2, 1)$

$\theta = \{x = 17\}$

$Mod(17, 2, 1)$

Generalized Modus Ponens

Definition

p_1'

p_2'

$\ldots$

p_n'

$p_1 \land p_2 \land \ldots \land p_n \Rightarrow q$

$\theta : \forall i \ \textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i)$

$\textsc{Subst}(\theta, q)$

Example

$Odd(17)$

$\forall x \ Odd(x) \Rightarrow Mod(x, 2, 1)$

$\theta = \{x = 17\}$

$Mod(17, 2, 1)$

Unification

Formally

$\textsc{Unify}(p, q) = \theta$ where $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$

Example

$p = \forall y \; Eats(Panda, y)$

$q =$	$\textsc{Unify}(p, q)$
$Eats(Panda, Leaves)$	$\{y/Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x/Panda, y/Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x/Panda, y/FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	

Unification

Formally

$\textsc{Unify}(p, q) = \theta$ where $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$

Example

$p = \forall y\ Eats(Panda, y)$

$q =$	$\textsc{Unify}(p, q)$
$Eats(Panda, Leaves)$	$\{y{=}Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x{=}Panda, y{=}Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x{=}Panda, y{=}FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	*fail*

Unification

Formally

$\textsc{Unify}(p, q) = \theta$ where $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$

Example

$p = \forall y\ Eats(Panda, y)$

$q =$	$\textsc{Unify}(p, q)$
$Eats(Panda, Leaves)$	$\{y{=}Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x{=}Panda, y{=}Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x{=}Panda, y{=}FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	*fail*

Unification

Formally

$\textsc{Unify}(p, q) = \theta$ where $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$

Example

$p = \forall y\ Eats(Panda, y)$

$q =$	$\textsc{Unify}(p, q)$
$Eats(Panda, Leaves)$	$\{y{=}Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x{=}Panda, y{=}Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x{=}Panda, y{=}FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	*fail*

Unification

Formally

$\textsc{Unify}(p, q) = \theta$ where $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$

Example

$p = \forall y\ Eats(Panda, y)$

$q =$	$\textsc{Unify}(p, q)$
$Eats(Panda, Leaves)$	$\{y{=}Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x{=}Panda, y{=}Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x{=}Panda, y{=}FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	*fail*

Unification

Formally

$\text{UNIFY}(p, q) = \theta$ where $\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$

Example

$p = \forall y\ Eats(Panda, y)$

$q =$	$\text{UNIFY}(p, q)$
$Eats(Panda, Leaves)$	$\{y{=}Leaves\}$
$\forall x Eats(x, Pizza)$	$\{x{=}Panda, y{=}Pizza\}$
$\forall x Eats(x, FavoriteFood(x))$	$\{x{=}Panda, y{=}FavoriteFood(Panda)\}$
$\forall x, y Eats(x, y) \Rightarrow Edible(y)$	*fail*

Forward Chaining

Key Ideas

- Repeatedly apply Generalized Modus Ponens
- Stop when nothing new can be inferred

Details

KB must be only first-order definite clauses, one of:

- Atomic clauses, e.g. $Mammal(Platypus)$
- Implications like $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$

Forward Chaining

Key Ideas

- Repeatedly apply Generalized Modus Ponens
- Stop when nothing new can be inferred

Details

KB must be only first-order definite clauses, one of:

- Atomic clauses, e.g. *Mammal*(*Platypus*)
- Implications like $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$

Forward Chaining Example

If you're rich and someone sells something you want, you buy it

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

If you're hot, you want ice cream

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

Brusters sells all kinds of ice cream

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

One flavor of ice cream is mint

$IceCream(Mint)$

Bill is rich

$Rich(Bill)$

Bill is hot

$Hot(Bill)$

Forward Chaining Example

$Hot(Bill)$
$IceCream(Mint)$
$\forall x \forall y \; Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$
$\theta = \{x = Bill, y = Mint\}$

Forward Chaining Example

$Hot(Bill)$
$IceCream(Mint)$
$\forall x \forall y \; Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$
$\theta = \{x = Bill, y = Mint\}$

$Wants(Bill, Mint)$

Forward Chaining Example

$$
\begin{array}{l}
Hot(Bill) \\
IceCream(Mint) \\
\forall x \forall y \; Hot(x) \land IceCream(y) \Rightarrow Wants(x, y) \\
\theta = \{x = Bill, y = Mint\} \\
\hline
Wants(Bill, Mint)
\end{array}
$$

$$
\begin{array}{l}
IceCream(Mint) \\
\forall z \; IceCream(z) \Rightarrow Sells(Brusters, z) \\
\theta = \{z = Mint\} \\
\hline
\end{array}
$$

Forward Chaining Example

$$\frac{\begin{array}{l} Hot(Bill) \\ IceCream(Mint) \\ \forall x \forall y\ Hot(x) \land IceCream(y) \Rightarrow Wants(x, y) \\ \theta = \{x = Bill, y = Mint\} \end{array}}{Wants(Bill, Mint)}$$

$$\frac{\begin{array}{l} IceCream(Mint) \\ \forall z\ IceCream(z) \Rightarrow Sells(Brusters, z) \\ \theta = \{z = Mint\} \end{array}}{Sells(Brusters, Mint)}$$

Forward Chaining Example

$Hot(Bill)$
$IceCream(Mint)$
$\forall x \forall y \; Hot(x) \land IceCream(y) \Rightarrow Wants(x, y)$
$\theta = \{x = Bill, y = Mint\}$

$Wants(Bill, Mint)$

$IceCream(Mint)$
$\forall z \; IceCream(z) \Rightarrow Sells(Brusters, z)$
$\theta = \{z = Mint\}$

$Sells(Brusters, Mint)$

$Rich(Bill)$
$Wants(Bill, Mint)$
$Sells(Brusters, Mint)$
$\forall u \forall v \forall w \; Rich(u) \land Wants(u, v) \land Sells(w, v) \Rightarrow Buys(u, v)$
$\theta = \{u = Bill, v = Mint, w = Brusters\}$

Forward Chaining Example

$Hot(Bill)$
$IceCream(Mint)$
$\forall x \forall y \; Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$
$\theta = \{x = Bill, y = Mint\}$

$Wants(Bill, Mint)$

$IceCream(Mint)$
$\forall z \; IceCream(z) \Rightarrow Sells(Brusters, z)$
$\theta = \{z = Mint\}$

$Sells(Brusters, Mint)$

$Rich(Bill)$
$Wants(Bill, Mint)$
$Sells(Brusters, Mint)$
$\forall u \forall v \forall w \; Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$
$\theta = \{u = Bill, v = Mint, w = Brusters\}$

$Buys(Bill, Mint)$

Forward Chaining Properties

Basic Properties

- Sound - uses Generalized Modus Ponens
- Complete - proof similar to propositional logic
- Works only with definite clauses

Termination

With no functions, p predicates, n constants, and at most k arguments per predicate:

Maximum Facts? pn^k

Maximum Iterations? pn^k

With f functions? May never terminate

Forward Chaining Properties

Basic Properties

- Sound - uses Generalized Modus Ponens
- Complete - proof similar to propositional logic
- Works only with definite clauses

Termination

With no functions, p predicates, n constants, and at most k arguments per predicate:

Maximum Facts? pn^k

Maximum Iterations? pn^k

With f functions? May never terminate

Forward Chaining Properties

Basic Properties

- Sound - uses Generalized Modus Ponens
- Complete - proof similar to propositional logic
- Works only with definite clauses

Termination

With no functions, p predicates, n constants, and at most k arguments per predicate:

Maximum Facts? pn^k

Maximum Iterations? pn^k

With f functions? May never terminate

Forward Chaining Properties

Basic Properties

- Sound - uses Generalized Modus Ponens
- Complete - proof similar to propositional logic
- Works only with definite clauses

Termination

With no functions, p predicates, n constants, and at most k arguments per predicate:

Maximum Facts? pn^k

Maximum Iterations? pn^k

With f functions? May never terminate

Forward Chaining Properties

Basic Properties

- Sound - uses Generalized Modus Ponens
- Complete - proof similar to propositional logic
- Works only with definite clauses

Termination

With no functions, p predicates, n constants, and at most k arguments per predicate:

Maximum Facts? pn^k

Maximum Iterations? pn^k

With f functions? May never terminate

Optimizing Forward Chaining

Indexing

- Treat facts like database relations
- Index by predicate + arguments
- Can get $O(1)$ fact retrieval
- Standard time/space tradeoffs

Rule Checking

- Don't check all rules on each iteration
- Check rules when new part of premise is satisfied

Backward Chaining

Key Ideas

- Start with the terms in the query
- Look for sentences that can conclude those terms using Generalized Modus Ponens
- Recurse as necessary to find simple terms

Details

- KB must be only first-order definite clauses

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad$ $Rich(Bill)$ $\qquad$ $Hot(Bill)$

Prove $Buys(a, b)$:

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w \; Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y \; Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z \; IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$
$[Hot(a), IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad Rich(Bill)$ $\qquad Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$
$[Hot(a), IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$
$[Hot(a), IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w\ Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y\ Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z\ IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad Rich(Bill)$ $\qquad Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$
$[Hot(a), IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint\}$
$[IceCream(z)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint, z{=}Brusters\}$

Backward Chaining Example

Given:

$\forall u \forall v \forall w \; Rich(u) \wedge Wants(u, v) \wedge Sells(w, v) \Rightarrow Buys(u, v)$

$\forall x \forall y \; Hot(x) \wedge IceCream(y) \Rightarrow Wants(x, y)$

$\forall z \; IceCream(z) \Rightarrow Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $Buys(a, b)$:

Goals	Assignment
$[Buys(a, b)]$	$\{\}$
$[Rich(a), Wants(a, b), Sells(z, b)]$	$\{u{=}a, v{=}b\}$
$[Wants(a, b), Sells(z, b)]$	$\{u{=}a{=}Bill, v{=}b\}$
$[Hot(a), IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[IceCream(b), Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y\}$
$[Sells(z, b)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint\}$
$[IceCream(z)]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint, z{=}Brusters\}$
$[]$	$\{u{=}a{=}Bill{=}x, v{=}b{=}y{=}Mint, z{=}Brusters\}$

Backward Chaining Properties

Properties

Sound?	Yes, uses Generalized Modus Ponens
Complete?	No, possible infinite loops
Space?	Linear in size of proof (depth-first search)

Cache proven goals:

- Complete for knowledge bases without functions
- Memory footprint increased

Backward Chaining Properties

Properties

Sound?	Yes, uses Generalized Modus Ponens
Complete?	No, possible infinite loops
Space?	Linear in size of proof (depth-first search)

Cache proven goals:

- Complete for knowledge bases without functions
- Memory footprint increased

Backward Chaining Properties

Properties

Sound?	Yes, uses Generalized Modus Ponens
Complete?	No, possible infinite loops
Space?	Linear in size of proof (depth-first search)

Cache proven goals:

- Complete for knowledge bases without functions
- Memory footprint increased

Backward Chaining Properties

Properties

- **Sound?** Yes, uses Generalized Modus Ponens
- **Complete?** No, possible infinite loops
- **Space?** Linear in size of proof (depth-first search)

Cache proven goals:

- Complete for knowledge bases without functions
- Memory footprint increased

Backward Chaining Properties

Properties

Sound?	Yes, uses Generalized Modus Ponens
Complete?	No, possible infinite loops
Space?	Linear in size of proof (depth-first search)

Cache proven goals:

- Complete for knowledge bases without functions
- Memory footprint increased

Prolog

Prolog Overview

- Backward chaining + many optimizations
- Millions of logical inferences per second
- "Database semantics"
  - Unique names
  - Closed world (not known to be true ⇒ false)
  - Domain closure (all constants given)

Prolog Syntax

```
factorial(1, 1).
factorial(N, _) :- N =< 0, fail.
factorial(N, F) :- N > 1, N1 is N - 1,
                   factorial(N1, F1), F is F1 * N.
```

Prolog

Prolog Overview

- Backward chaining + many optimizations
- Millions of logical inferences per second
- “Database semantics”
    - Unique names
    - Closed world (not known to be true ⇒ false)
    - Domain closure (all constants given)

Prolog Syntax

```
factorial(1, 1).
factorial(N, _) :- N =< 0, fail.
factorial(N, F) :- N > 1, N1 is N - 1,
                   factorial(N1, F1), F is F1 * N.
```

Prolog Examples

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z\ Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z\ Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$$\mathit{Connected}(A, F)$$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$\mathit{Connected}(A, F)$

$\mathit{Edge}(A, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, F)$

fail

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$$\mathit{Connected}(A, F)$$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$\mathit{Connected}(A, F)$

$\mathit{Edge}(A, y)$ $\qquad$ $\mathit{Connected}(y, F)$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$\mathit{Connected}(A, F)$

$\mathit{Edge}(A, y)$ $\qquad\qquad$ $\mathit{Connected}(y, F)$

$\mathit{Edge}(A, B)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\quad$ $Connected(B, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z\ Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z\ Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\qquad$ $Connected(B, F)$

$Edge(B, F)$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

$\mathit{Connected}(A, F)$

$\mathit{Edge}(A, B)$ $\quad$ $\mathit{Connected}(B, F)$

$\mathit{Edge}(B, F)$

fail

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z\ Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z\ Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\quad\quad$ $Connected(B, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$$Connected(A, F)$$

$$Edge(A, B) \qquad Connected(B, F)$$

$$Edge(B, y) \qquad Connected(y, F)$$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

- $\mathit{Connected}(A, F)$
  - $\mathit{Edge}(A, B)$
  - $\mathit{Connected}(B, F)$
    - $\mathit{Edge}(B, y)$
      - $\mathit{Edge}(B, C)$
    - $\mathit{Connected}(y, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\quad$ $Connected(B, F)$

$Edge(B, C)$ $\quad$ $Connected(C, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

- $Connected(A, F)$
  - $Edge(A, B)$
  - $Connected(B, F)$
    - $Edge(B, C)$
    - $Connected(C, F)$
      - $Edge(C, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

- $Connected(A, F)$
  - $Edge(A, B)$
  - $Connected(B, F)$
    - $Edge(B, C)$
    - $Connected(C, F)$
      - $Edge(C, F)$
        - fail

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

- $Connected(A, F)$
  - $Edge(A, B)$
  - $Connected(B, F)$
    - $Edge(B, C)$
    - $Connected(C, F)$

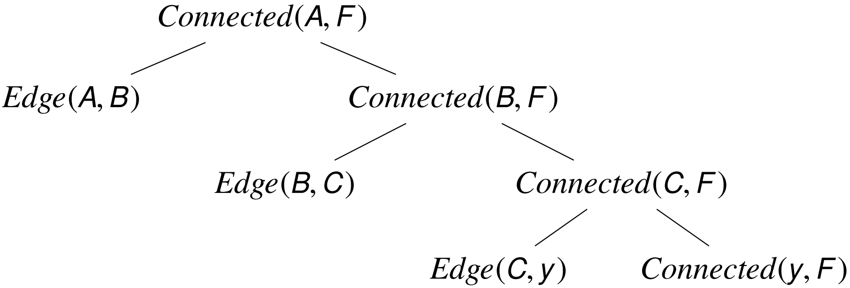
Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$



Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

- $Connected(A, F)$
  - $Edge(A, B)$
  - $Connected(B, F)$
    - $Edge(B, C)$
    - $Connected(C, F)$
      - $Edge(C, y)$
        - fail
      - $Connected(y, F)$

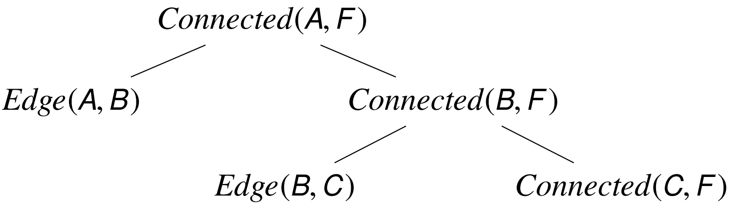
Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$



Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z\ Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z\ Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\quad$ $Connected(B, F)$

$Edge(B, y)$ $\quad$ $Connected(y, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z\ Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z\ Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

- $Connected(A, F)$
  - $Edge(A, B)$
  - $Connected(B, F)$
    - $Edge(B, y)$
      - $Edge(B, D)$
    - $Connected(y, F)$

Backward Chaining Backtracking

Prove: $\mathit{Connected}(A, F)$

Given: $\forall x, z\ \mathit{Edge}(x, z) \Rightarrow \mathit{Connected}(x, z)$

$\forall x, y, z\ \mathit{Edge}(x, y) \wedge \mathit{Connected}(y, z) \Rightarrow \mathit{Connected}(x, z)$

$\mathit{Edge}(A, B) \wedge \mathit{Edge}(B, C) \wedge \mathit{Edge}(B, D) \wedge \mathit{Edge}(D, F)$

- $\mathit{Connected}(A, F)$
  - $\mathit{Edge}(A, B)$
  - $\mathit{Connected}(B, F)$
    - $\mathit{Edge}(B, D)$
    - $\mathit{Connected}(D, F)$

Backward Chaining Backtracking

Prove: $Connected(A, F)$

Given: $\forall x, z \; Edge(x, z) \Rightarrow Connected(x, z)$

$\forall x, y, z \; Edge(x, y) \wedge Connected(y, z) \Rightarrow Connected(x, z)$

$Edge(A, B) \wedge Edge(B, C) \wedge Edge(B, D) \wedge Edge(D, F)$

$Connected(A, F)$

$Edge(A, B)$ $\quad$ $Connected(B, F)$

$Edge(B, D)$ $\quad$ $Connected(D, F)$

$Edge(D, F)$

Prolog Example

Resolution

Definition

$$\frac{\begin{array}{l} p_1 \vee \ldots \vee p_n \\ q_1 \vee \ldots \vee q_m \\ \theta : \textsc{Subst}(\theta, p_i) = \neg\textsc{Subst}(\theta, q_j) \end{array}}{\begin{array}{l} \textsc{Subst}(\theta, p_1 \vee \ldots p_{i-1} \vee p_{i+1} \ldots \vee p_n \\ \qquad\qquad \vee\, q_1 \vee \ldots q_{j-1} \vee q_{j+1} \ldots \vee q_m) \end{array}}$$

Example

$$\frac{\begin{array}{l} \neg Mammal(x) \vee WarmBlooded(x) \\ Mammal(Platypus) \\ \theta = \{x = Platypus\} \end{array}}{WarmBlooded(Platypus)}$$

Resolution

Definition

$$\frac{\begin{array}{l} p_1 \lor \ldots \lor p_n \\ q_1 \lor \ldots \lor q_m \\ \theta : \textsc{Subst}(\theta, p_i) = \neg\textsc{Subst}(\theta, q_j) \end{array}}{\begin{array}{l} \textsc{Subst}(\theta, p_1 \lor \ldots p_{i-1} \lor p_{i+1} \ldots \lor p_n \\ \qquad\qquad \lor q_1 \lor \ldots q_{j-1} \lor q_{j+1} \ldots \lor q_m) \end{array}}$$

Example

$$\frac{\begin{array}{l} \neg Mammal(x) \lor WarmBlooded(x) \\ Mammal(Platypus) \\ \theta = \{x = Platypus\} \end{array}}{WarmBlooded(Platypus)}$$

Resolution

Resolution Procedure

1. Convert knowledge base to CNF
2. Convert query to CNF
3. Assume ¬ query
4. Apply resolution until *false* is concluded

CNF Complications

- Negations moved through ∀ and ∃
- All quantifiers must have different variable names
- Quantifier scopes handled through skolemization

Resolution

Resolution Procedure

1. Convert knowledge base to CNF
2. Convert query to CNF
3. Assume $\neg$ query
4. Apply resolution until *false* is concluded

CNF Complications

- Negations moved through $\forall$ and $\exists$
- All quantifiers must have different variable names
- Quantifier scopes handled through skolemization

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

Resolution Example

Given:

$\forall u \forall v \forall w \; \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y \; \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y \; \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y \; Buys(a, b)$:

$\forall a \forall b \; \neg Buys(a, b)$

Resolution Example

Given:

$\forall u \forall v \forall w \; \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y \; \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y \; \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y \; Buys(a, b)$:

$\forall a \forall b \; \neg Buys(a, b)$

$\forall u \forall v \forall w \; \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u,v) \vee \neg Sells(w,v) \vee Buys(u,v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x,y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a,b)$:

$\forall a \forall b\ \neg Buys(a,b)$

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u,v) \vee \neg Sells(w,v) \vee Buys(u,v)$

$\theta = \{a = u, b = v\}$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$$\frac{\begin{array}{l}\forall a \forall b\ \neg Buys(a, b) \\ \forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v) \\ \theta = \{a = u, b = v\}\end{array}}{\forall a \forall b \forall w\ \neg Rich(a) \vee \neg Wants(a, b) \vee \neg Sells(w, b)}$$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad Rich(Bill)$ $\qquad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \vee \neg Wants(a, b) \vee \neg Sells(w, b)$

$Rich(Bill)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \vee \neg Wants(a, b) \vee \neg Sells(w, b)$

$Rich(Bill)$

$\theta = \{a = u = Bill, b = v\}$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad Rich(Bill)$ $\qquad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \vee \neg Wants(a, b) \vee \neg Sells(w, b)$

$Rich(Bill)$

$\theta = \{a = u = Bill, b = v\}$

$\forall b \forall w\ \neg Wants(Bill, b) \vee \neg Sells(w, b)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

$\forall y\ \neg IceCream(z) \lor Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \lor \neg Wants(a, b) \lor \neg Sells(w, b)$

$Rich(Bill)$

$\theta = \{a = u = Bill, b = v\}$

$\forall b \forall w\ \neg Wants(Bill, b) \lor \neg Sells(w, b)$

$\forall x \forall y\ \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \vee \neg Wants(a, b) \vee \neg Sells(w, b)$

$Rich(Bill)$

$\theta = \{a = u = Bill, b = v\}$

$\forall b \forall w\ \neg Wants(Bill, b) \vee \neg Sells(w, b)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\theta = \{a = u = Bill = x, b = v = y\}$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

$\forall y\ \neg IceCream(z) \lor Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\forall u \forall v \forall w\ \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\theta = \{a = u, b = v\}$

$\forall a \forall b \forall w\ \neg Rich(a) \lor \neg Wants(a, b) \lor \neg Sells(w, b)$

$Rich(Bill)$

$\theta = \{a = u = Bill, b = v\}$

$\forall b \forall w\ \neg Wants(Bill, b) \lor \neg Sells(w, b)$

$\forall x \forall y\ \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

$\theta = \{a = u = Bill = x, b = v = y\}$

$\forall b \forall w\ \neg Hot(Bill) \lor \neg IceCream(b) \lor \neg Sells(w, b)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint) \qquad Rich(Bill) \qquad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$$\frac{\ldots}{\forall b \forall w\ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w, b)}$$

Resolution Example

Given:

$\forall u \forall v \forall w \; \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\forall x \forall y \; \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

$\forall y \; \neg IceCream(z) \lor Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y \; Buys(a, b)$:

$\forall a \forall b \; \neg Buys(a, b)$

$\ldots$

$\forall b \forall w \; \neg Hot(Bill) \lor \neg IceCream(b) \lor \neg Sells(w, b)$

$\forall y \; \neg IceCream(z) \lor Sells(Brusters, z)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\ldots$

$\forall b \forall w\ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w, b)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

Resolution Example

Given:

$\forall u \forall v \forall w \ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y \ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y \ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y \ Buys(a, b)$:

$\forall a \forall b \ \neg Buys(a, b)$

$\ldots$

$\forall b \forall w \ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w, b)$

$\forall y \ \neg IceCream(z) \vee Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

$\forall y \ \neg Hot(Bill) \vee \neg IceCream(b)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\qquad$ $Rich(Bill)$ $\qquad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\ldots$

$\forall b \forall w\ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w, b)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

$\forall y\ \neg Hot(Bill) \vee \neg IceCream(b)$

$IceCream(Mint)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u,v) \vee \neg Sells(w,v) \vee Buys(u,v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x,y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a,b)$:

$\forall a \forall b\ \neg Buys(a,b)$

$\ldots$

$\forall b \forall w\ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w,b)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

$\forall y\ \neg Hot(Bill) \vee \neg IceCream(b)$

$IceCream(Mint)$

$\theta = \{a = u = Bill = x, b = v = y = z = Mint, w = Brusters\}$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \vee \neg Wants(u, v) \vee \neg Sells(w, v) \vee Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \vee \neg IceCream(y) \vee Wants(x, y)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$IceCream(Mint)$ $\quad Rich(Bill)$ $\quad Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\ldots$

$\forall b \forall w\ \neg Hot(Bill) \vee \neg IceCream(b) \vee \neg Sells(w, b)$

$\forall y\ \neg IceCream(z) \vee Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

$\forall y\ \neg Hot(Bill) \vee \neg IceCream(b)$

$IceCream(Mint)$

$\theta = \{a = u = Bill = x, b = v = y = z = Mint, w = Brusters\}$

$\neg Hot(Bill)$

Resolution Example

Given:

$\forall u \forall v \forall w\ \neg Rich(u) \lor \neg Wants(u, v) \lor \neg Sells(w, v) \lor Buys(u, v)$

$\forall x \forall y\ \neg Hot(x) \lor \neg IceCream(y) \lor Wants(x, y)$

$\forall y\ \neg IceCream(z) \lor Sells(Brusters, z)$

$IceCream(Mint)$ $\quad$ $Rich(Bill)$ $\quad$ $Hot(Bill)$

Prove $\exists x \exists y\ Buys(a, b)$:

$\forall a \forall b\ \neg Buys(a, b)$

$\ldots$

$\forall b \forall w\ \neg Hot(Bill) \lor \neg IceCream(b) \lor \neg Sells(w, b)$

$\forall y\ \neg IceCream(z) \lor Sells(Brusters, z)$

$\theta = \{a = u = Bill = x, b = v = y = z, w = Brusters\}$

$\forall y\ \neg Hot(Bill) \lor \neg IceCream(b)$

$IceCream(Mint)$

$\theta = \{a = u = Bill = x, b = v = y = z = Mint, w = Brusters\}$

$\neg Hot(Bill)$

$Hot(Bill)$

Resolution Properties

Properties

Sound? Yes, uses Resolution

Complete? Yes, with subset resolution or factoring

Retrieving existential goals may require adding $Answer(x)$

Resolution Properties

Properties

Sound? Yes, uses Resolution

Complete? Yes, with subset resolution or factoring

Retrieving existential goals may require adding $Answer(x)$

Resolution Properties

Properties

Sound? Yes, uses Resolution

Complete? Yes, with subset resolution or factoring

Retrieving existential goals may require adding $Answer(x)$

Resolution Properties

Properties

Sound?	Yes, uses Resolution
Complete?	Yes, with subset resolution or factoring

Retrieving existential goals may require adding $Answer(x)$

Key Ideas

Translating First-Order Logic

- Identify objects (terms) and relations (predicates)
- Generally, use $\Rightarrow$ with $\forall$ and $\wedge$ with $\exists$
- Use inequality to specify unique objects

First-Order Logic Inference

- Forward chaining on definite clauses is sound and complete
- Backward chaining on definite clauses is sound
- Resolution is sound and complete