# CSC 481: Resolution

## 1- Motivation, the Propositional case


**Rodrigo Canaan**
**Assistant Professor**
**Computer Science Department**
**Cal Poly, San Luis Obispo**
**rcanaan@calpoly.edu**

# Motivation

# Motivation

- So far, we have done inference “by hand”

# Motivation

- So far, we have done inference “by hand”
- We would like to have a way to mechanize this process!

# Motivation

# Motivation

- Many arguments we used informally were of the form:

# Motivation

- Many arguments we used informally were of the form:

*“if a  or b  must be true, but a isn’t, then b  must be”*

# Motivation

- Many arguments we used informally were of the form:

*"if $a$ or $b$ must be true, but $a$ isn't, then $b$ must be"*

- This is a simplified form of a rule of inference called **resolution**

# Motivation

- Many arguments we used informally were of the form:

  *“if $a$ or $b$ must be true, but $a$ isn’t, then $b$ must be”*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

# Motivation

- Many arguments we used informally were of the form:

*"if a  or b  must be true, but a isn't, then b  must be"*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$A \vee B$$

# Motivation

- Many arguments we used informally were of the form:

*"if a  or b  must be true, but a isn't, then b  must be"*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$A \vee B$$

$$\neg A$$

# Motivation

- Many arguments we used informally were of the form:

*"if a or b must be true, but a isn't, then b must be"*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$\frac{\begin{array}{l} A \lor B \\ \neg A \end{array}}{B}$$

# Motivation

- Many arguments we used informally were of the form:

*"if a or b must be true, but a isn't, then b must be"*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$\frac{\begin{array}{c} A \vee B \\ \neg A \end{array}}{B}$$

- This will form the basis of our proof method!

# Motivation

- Many arguments we used informally were of the form:

*“if a or b must be true, but a isn’t, then b must be”*

- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$\frac{\begin{array}{c} A \lor B \\ \neg A \end{array}}{B}$$

- This will form the basis of our proof method!

# Motivation

- Many arguments we used informally were of the form:

*"if a or b must be true, but a isn't, then b must be"*

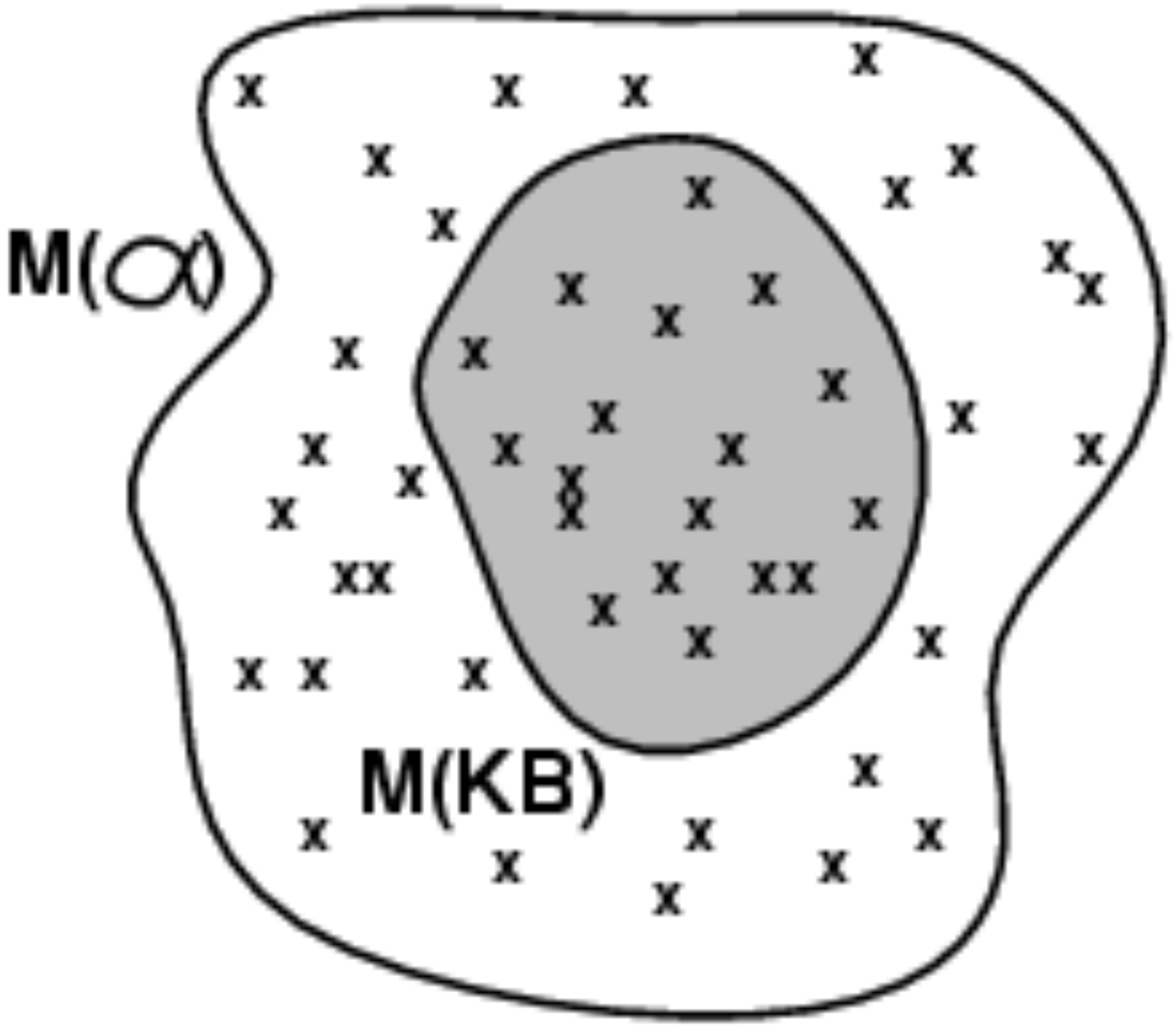
- This is a simplified form of a rule of inference called **resolution**
- Can be seen as taking two sentences and returning a third sentence:

$$\frac{\begin{array}{c} A \vee B \\ \neg A \end{array}}{B}$$

Also written $A \vee B, \neg A \vdash B$

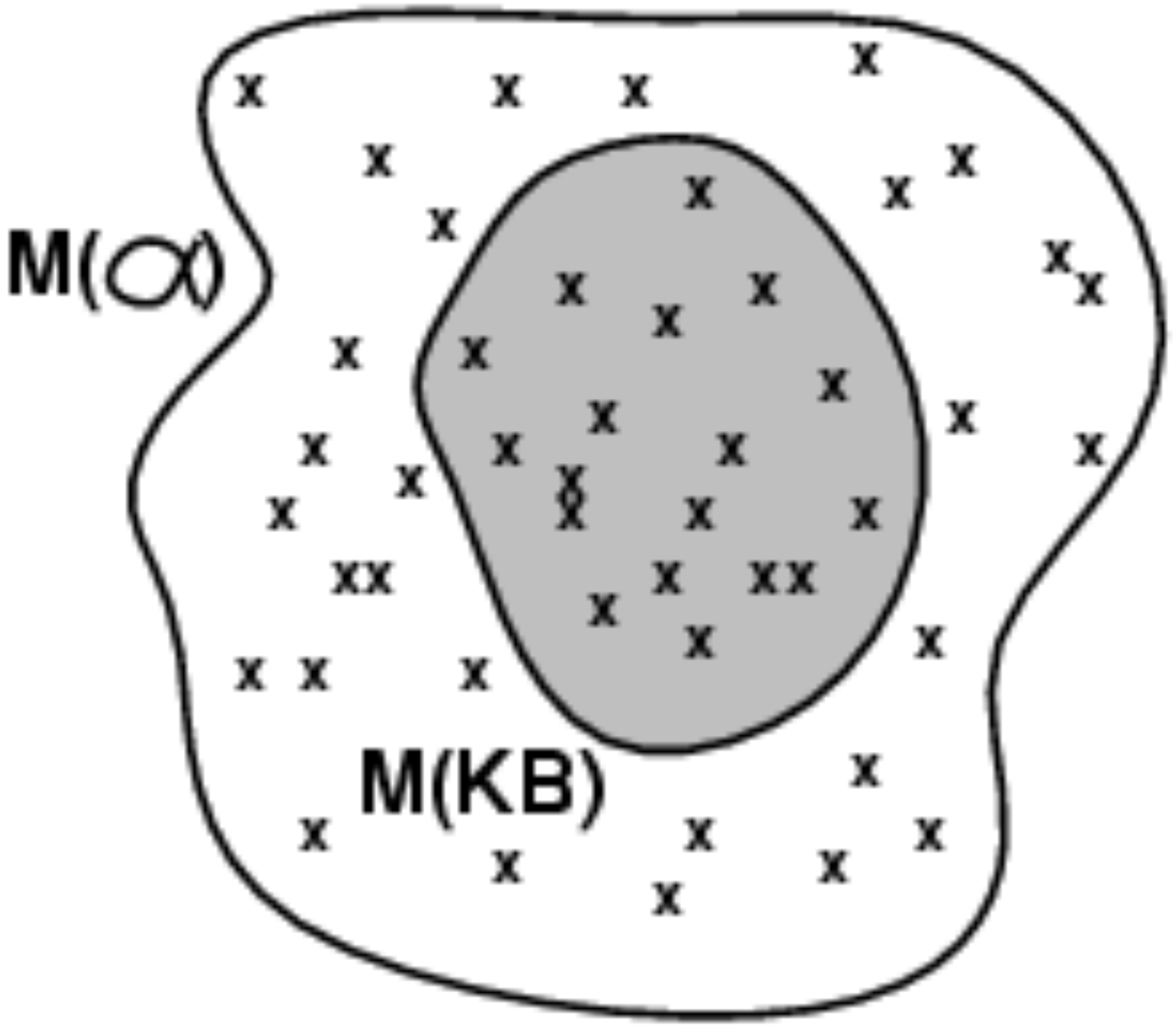
- This will form the basis of our proof method!

# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

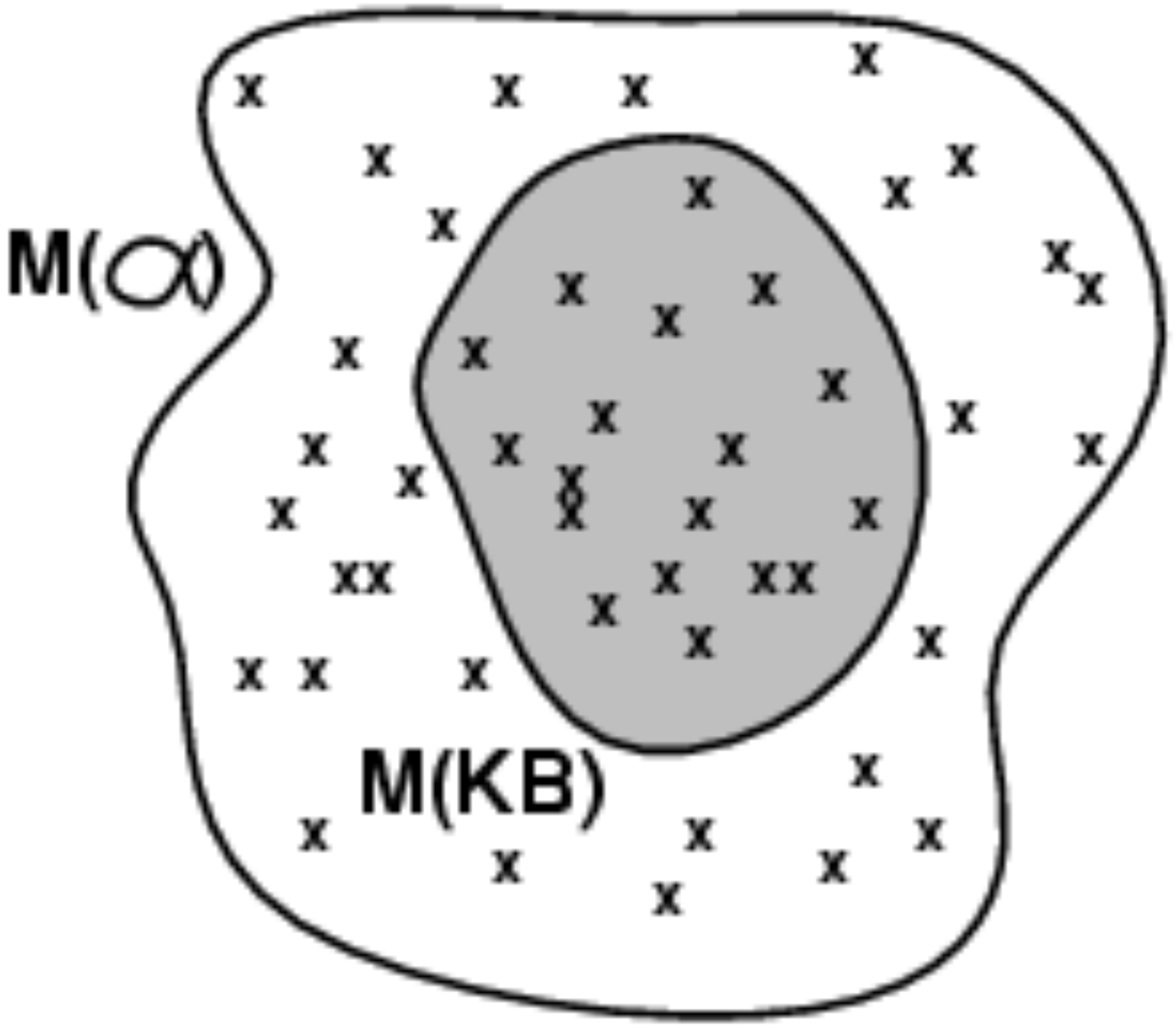
# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

- $KB \vDash \alpha$ means that, in all interpretations (or models) where KB is true, $\alpha$ must also be

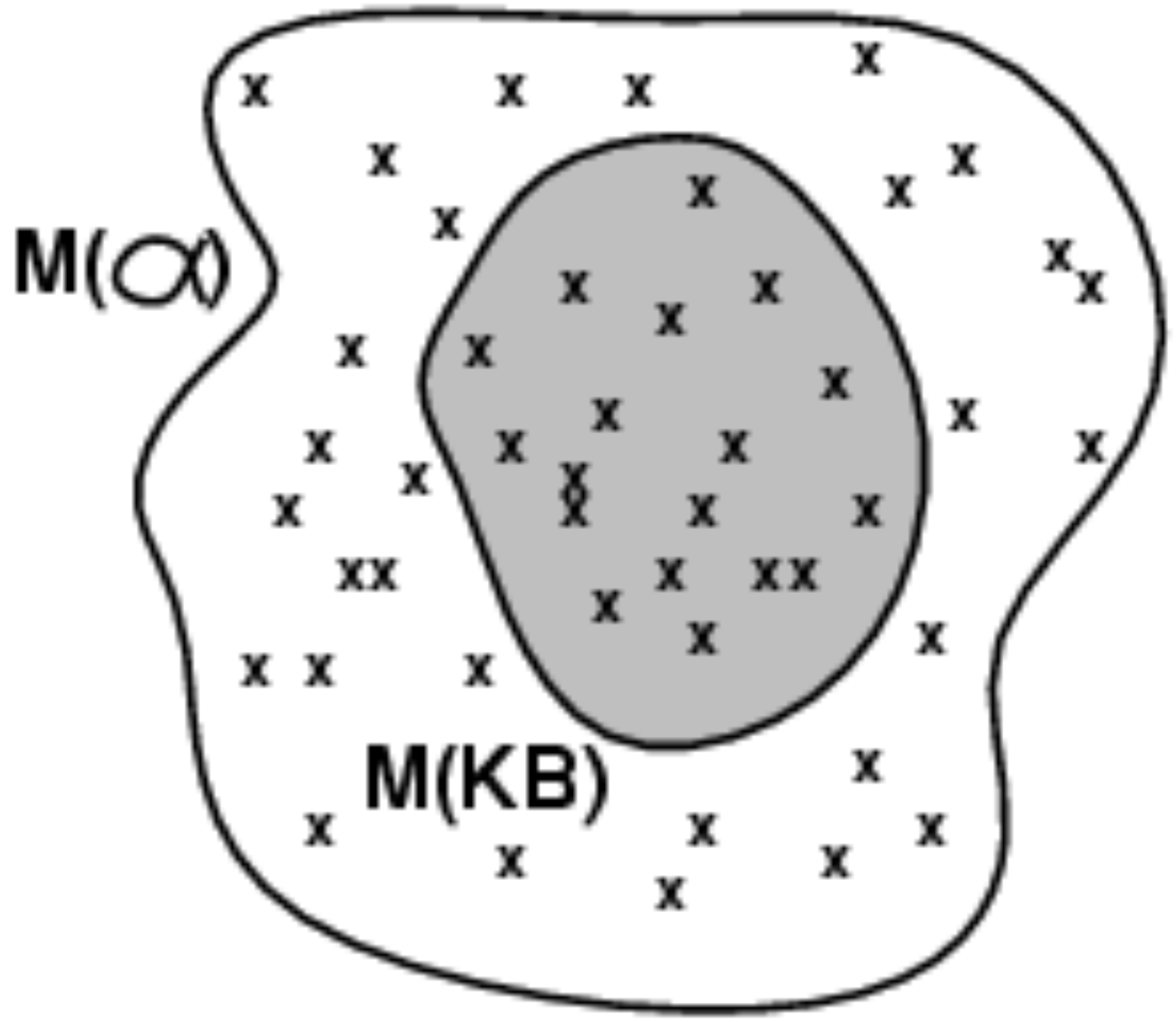
# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

- $KB \vDash \alpha$ means that, in all interpretations (or models) where KB is true, $\alpha$ must also be
- KB is a *stronger assertion* than $\alpha$

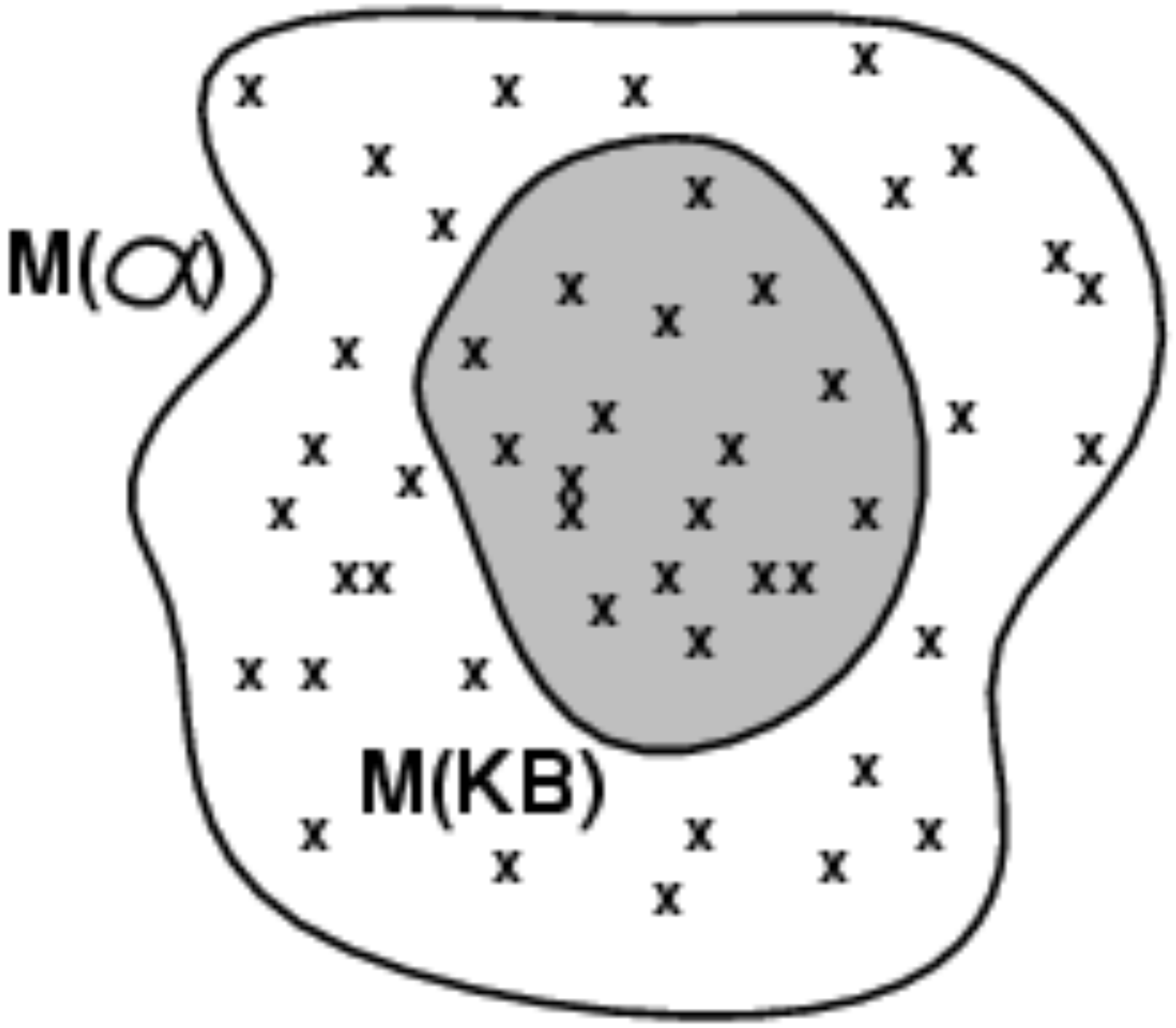
# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

- $KB \models \alpha$ means that, in all interpretations (or models) where KB is true, $\alpha$ must also be
- KB is a *stronger assertion* than $\alpha$
- Example:
  - KB = “The Mustangs won by 10 points”
  - $\alpha$ = “The Mustangs won”

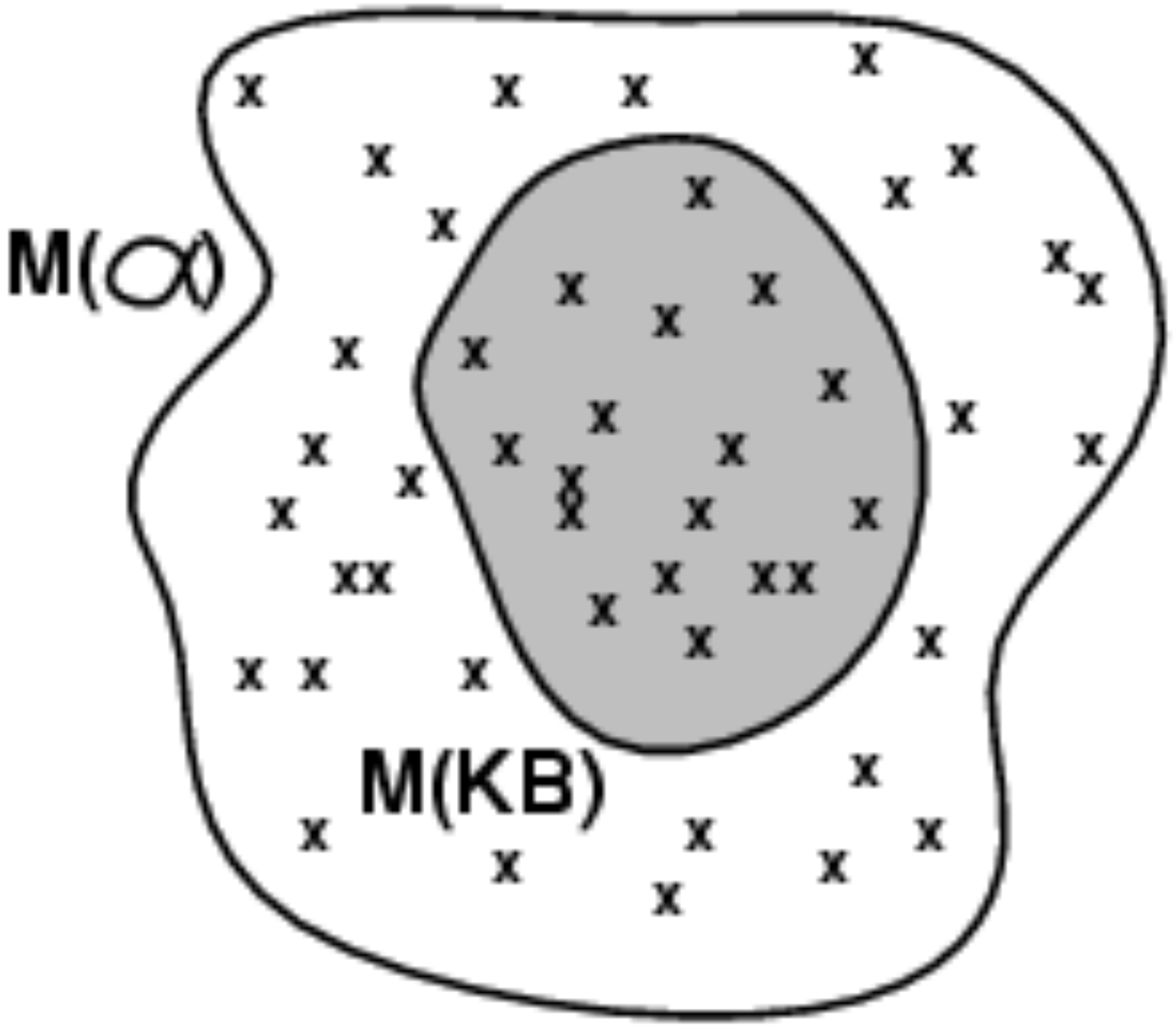
# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

- $KB \vDash \alpha$ means that, in all interpretations (or models) where KB is true, $\alpha$ must also be
- KB is a *stronger assertion* than $\alpha$
- Example:
  - KB = “The Mustangs won by 10 points”
  - $\alpha$ = “The Mustangs won”
- **“Proving something” means showing that it is entailed by our assumptions**

# Remember entailment




http://aima.eecs.berkeley.edu/slides-ppt/

- $KB \vDash \alpha$ means that, in all interpretations (or models) where KB is true, $\alpha$ must also be
- KB is a *stronger assertion* than $\alpha$
- Example:
  - KB = “The Mustangs won by 10 points”
  - $\alpha$ = “The Mustangs won”
- **“Proving something” means showing that it is entailed by our assumptions**
- But doing it by explicitly enumerating interpretations is infeasible

# Remember Entailment

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

  1. KB $\models \alpha$ (read "**S entails** $\alpha$")

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

  1. KB $\models \alpha$ (read "**S entails** $\alpha$")

  2. For every $\mathfrak{I}$, if $\mathfrak{I} \models$ KB, then $\mathfrak{I} \models \alpha$

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

  1. KB $\models \alpha$ (read "**S entails** $\alpha$")

  2. For every $\mathfrak{I}$, if $\mathfrak{I} \models$ KB, then $\mathfrak{I} \models \alpha$

  3. There is no $\mathfrak{I}$ such that $\mathfrak{I} \models$ KB $\cup \{\neg\alpha\}$

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

  1. KB $\models \alpha$ (read "**S entails** $\alpha$")

  2. For every $\mathfrak{I}$, if $\mathfrak{I} \models$ KB, then $\mathfrak{I} \models \alpha$

  3. There is no $\mathfrak{I}$ such that $\mathfrak{I} \models$ KB $\cup \{\neg\alpha\}$

  4. KB $\cup \{\neg\alpha\}$ is unsatisfiable

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

1. KB $\models \alpha$ (read "**S entails** $\alpha$")

2. For every $\mathfrak{I}$, if $\mathfrak{I} \models$ KB, then $\mathfrak{I} \models \alpha$

3. There is no $\mathfrak{I}$ such that $\mathfrak{I} \models$ KB $\cup \{\neg\alpha\}$

4. KB $\cup \{\neg\alpha\}$ is unsatisfiable

5. For every $\mathfrak{I}$, $\mathfrak{I} \models$ KB $\rightarrow \{\alpha\}$

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

1. KB $\models \alpha$ (read "**S entails** $\alpha$")
2. For every $\mathfrak{I}$, if $\mathfrak{I} \models$ KB, then $\mathfrak{I} \models \alpha$
3. There is no $\mathfrak{I}$ such that $\mathfrak{I} \models$ KB $\cup \{\neg\alpha\}$
4. KB $\cup \{\neg\alpha\}$ is unsatisfiable
5. For every $\mathfrak{I}$, $\mathfrak{I} \models$ KB $\rightarrow \{\alpha\}$
6. KB $\rightarrow \{\alpha\}$ is valid (i.e. a tautology)

# Remember Entailment

- Given a (finite) set of sentences KB and a sentence $\alpha$, the following statements are equivalent:

1. KB $\models \alpha$ (read "**S entails** $\alpha$")
2. For every $\Im$, if $\Im \models$ KB, then $\Im \models \alpha$
3. There is no $\Im$ such that $\Im \models$ KB $\cup \{\neg\alpha\}$
4. KB $\cup \{\neg\alpha\}$ is unsatisfiable
5. For every $\Im$, $\Im \models$ KB $\rightarrow \{\alpha\}$
6. KB $\rightarrow \{\alpha\}$ is valid (i.e. a tautology)

In 5 and 6, we abuse notation and interpret KB as the conjunction of all its sentences

# Idea

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.
- Express your question (query) as a sentence $\alpha$

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.
- Express your question (query) as a sentence $\alpha$
- Our goal becomes to prove $KB \models \alpha$

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.
- Express your question (query) as a sentence $\alpha$
- Our goal becomes to prove $KB \vDash \alpha$
- Use inference rules to generate new sentences and from existing sentences and add them to the KB

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.
- Express your question (query) as a sentence $\alpha$
- Our goal becomes to prove $KB \vDash \alpha$
- Use inference rules to generate new sentences and from existing sentences and add them to the KB
    - If you generate $\alpha$, return "true"!

# Idea

- Express your knowledge (KB) as a set of sentences $S_1, S_2, S_3$ etc.
- Express your question (query) as a sentence $\alpha$
- Our goal becomes to prove $KB \vDash \alpha$
- Use inference rules to generate new sentences and from existing sentences and add them to the KB
    - If you generate $\alpha$, return “true”!
    - If you’ve ran out of rules to apply return “false”!

# Idea

## Proof by Contradiction

# Idea

## Proof by Contradiction

- **Alternatively:**

# Idea

## Proof by Contradiction

- **Alternatively:**
  - Add $\neg\alpha$ (the negated query) to the KB

# Idea

## Proof by Contradiction

- **Alternatively:**
  - Add $\neg\alpha$ (the negated query) to the KB
  - Our goal becomes to prove that $KB \cup \neg\alpha$ is a contradiction

# Idea

## Proof by Contradiction

- **Alternatively:**
  - Add $\neg\alpha$ (the negated query) to the KB
  - Our goal becomes to prove that $KB \cup \neg\alpha$ is a contradiction
  - Use inference rules to generate new sentences and from existing sentences and add them to the KB

# Idea

## Proof by Contradiction

- **Alternatively:**
  - Add $\neg\alpha$ (the negated query) to the KB
  - Our goal becomes to prove that $KB \cup \neg\alpha$ is a contradiction
  - Use inference rules to generate new sentences and from existing sentences and add them to the KB
    - If you generate a contradiction, return “true”

# Idea

## Proof by Contradiction

- **Alternatively:**
  - Add $\neg\alpha$ (the negated query) to the KB
  - Our goal becomes to prove that $KB \cup \neg\alpha$ is a contradiction
  - Use inference rules to generate new sentences and from existing sentences and add them to the KB
    - If you generate a contradiction, return "true"
    - If you've ran out of rules to apply, return "false"!

# Soundness and completeness

## Proof by Contradiction

# Soundness and completeness

## Proof by Contradiction

- If whenever an algorithm returns “true”, we observe that $KB \models \alpha$, we say that the algorithm is **sound**

# Soundness and completeness

## Proof by Contradiction

- If whenever an algorithm returns “true”, we observe that $KB \models \alpha$, we say that the algorithm is **sound**
- If whenever $KB \models \alpha$, the algorithm returns “true”, we say that the algorithm is **complete**

# Good news and bad news

# Good news and bad news

**The bad news:**

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

**The good news:**

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

**The good news:**

- There exists an inference procedure that uses resolution as its *only* rule of inference.

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

**The good news:**

- There exists an inference procedure that uses resolution as its *only* rule of inference.
- This procedure is sound and, for the Propositional case, complete.

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

**The good news:**

- There exists an inference procedure that uses resolution as its *only* rule of inference.
- This procedure is sound and, for the Propositional case, complete.
- It is also *refutation complete in FOL*: will always terminate in a finite number of steps if a sentence is unsatisfiable.

# Good news and bad news

**The bad news:**

- There can be *no procedure* that is both sound and complete to determine if an arbitrary sentence is entailed by a set of FOL sentences.

**The good news:**

- There exists an inference procedure that uses resolution as its *only* rule of inference.
- This procedure is sound and, for the Propositional case, complete.
- It is also *refutation complete in FOL*: will always terminate in a finite number of steps if a sentence is unsatisfiable.

# Resolution in Propositional Logic

# Intuition

## Propositional case (no quantifiers, predicates or functions)

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies
- I will not go to the beach

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies
- I will not go to the beach
- Therefore, I will go to the movies

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies
- I will not go to the beach
- Therefore, I will go to the movies

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies $A \vee B$
- I will not go to the beach
- Therefore, I will go to the movies

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies $A \lor B$
- I will not go to the beach $\neg A$
- Therefore, I will go to the movies

# Intuition

## Propositional case (no quantifiers, predicates or functions)

- I will go to the beach or to the movies
- I will not go to the beach
- Therefore, I will go to the movies

$$\begin{array}{l} \cancel{A} \lor B \\ \cancel{\neg A} \\ \hline B \end{array}$$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
    - $S_1 = x \vee p_1 \vee p_2 \vee \ldots \vee p_n$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = x \vee p_1 \vee p_2 \vee \ldots \vee p_n$
  - $S_2 = \neg x \vee q_1 \vee q_2 \vee \ldots \vee q_m$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = x \vee p_1 \vee p_2 \vee \ldots \vee p_n$
  - $S_2 = \neg x \vee q_1 \vee q_2 \vee \ldots \vee q_m$
- We can derive:

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = x \lor p_1 \lor p_2 \lor \ldots \lor p_n$
  - $S_2 = \neg x \lor q_1 \lor q_2 \lor \ldots \lor q_m$
- We can derive:
  - $S_3 = p_1 \lor p_2 \lor \ldots \lor p_n \lor q_1 \lor q_2 \lor \ldots \lor q_m$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
    - $S_1 = x \vee p_1 \vee p_2 \vee \ldots \vee p_n$
    - $S_2 = \neg x \vee q_1 \vee q_2 \vee \ldots \vee q_m$
- We can derive:
    - $S_3 = p_1 \vee p_2 \vee \ldots \vee p_n \vee q_1 \vee q_2 \vee \ldots \vee q_m$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = \cancel{x} \vee p_1 \vee p_2 \vee \ldots \vee p_n$
  - $S_2 = \cancel{\neg x} \vee q_1 \vee q_2 \vee \ldots \vee q_m$
- We can derive:
  - $S_3 = p_1 \vee p_2 \vee \ldots \vee p_n \vee q_1 \vee q_2 \vee \ldots \vee q_m$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = \cancel{x} \vee \boxed{p_1 \vee p_2 \vee \ldots \vee p_n}$
  - $S_2 = \cancel{\neg x} \vee q_1 \vee q_2 \vee \ldots \vee q_m$
- We can derive:
  - $S_3 = \boxed{p_1 \vee p_2 \vee \ldots \vee p_n} \vee q_1 \vee q_2 \vee \ldots \vee q_m$

# The resolution rule

## Propositional case (no quantifiers, predicates or functions)

- More generically:
- Given two sentences in *Conjunctive Normal Form* (see next slide)
  - $S_1 = \cancel{x} \vee \boxed{p_1 \vee p_2 \vee \ldots \vee p_n}$
  - $S_2 = \cancel{\neg x} \vee \boxed{q_1 \vee q_2 \vee \ldots \vee q_m}$
- We can derive:
  - $S_3 = \boxed{p_1 \vee p_2 \vee \ldots \vee p_n} \vee \boxed{q_1 \vee q_2 \vee \ldots \vee q_m}$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach
  - Therefore, I will have Ice cream

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach
  - Therefore, I will have Ice cream

$$A \to B$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach
  - Therefore, I will have Ice cream

$$A \rightarrow B$$

$$A$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach
  - Therefore, I will have Ice cream

$$\frac{\begin{array}{l}\cancel{A} \to B \\ \cancel{A}\end{array}}{B}$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Modus Ponens is another commonly used rule
- Example:
  - If I go to the beach, i will have ice cream
  - I will go to the beach
  - Therefore, I will have Ice cream
- Exercise: prove that Modus Ponens is a special case of resolution

$$\frac{\cancel{A} \rightarrow B \qquad \cancel{A}}{B}$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive
- That is, if $A \to B$ and $B \to C$, then $A \to C$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive
- That is, if $A \to B$ and $B \to C$, then $A \to C$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive
- That is, if $A \to B$ and $B \to C$, then $A \to C$

$$A \to B$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive

- That is, if $A \to B$ and $B \to C$, then $A \to C$

$$A \to B$$

$$B \to C$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive

- That is, if $A \to B$ and $B \to C$, then $A \to C$

$$A \to B \qquad\qquad \neg A \vee \cancel{B}$$

$$B \to C \qquad\qquad \cancel{\phantom{B}}$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive
- That is, if $A \to B$ and $B \to C$, then $A \to C$

$$A \to B \qquad \neg A \vee \cancel{B}$$

$$B \to C \qquad \cancel{\neg B} \vee C$$

# Resolution and *Modus Ponens*

## Propositional case (no quantifiers, predicates or functions)

- Resolution can also be seen as a consequence of the fact that implication is transitive

- That is, if $A \to B$ and $B \to C$, then $A \to C$

$$A \to B \qquad\qquad \neg A \vee \cancel{B}$$

$$B \to C \qquad\qquad \cancel{\neg B} \vee C$$

$$\frac{\phantom{\neg A \vee C}}{\neg A \vee C}$$

# Discarding Tautologies

Is a tautology

- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the clause contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set *S*, it is added to *S*, and is considered for further resolution inferences.

Source: https://en.wikipedia.org/wiki/Resolution_(logic)

# Discarding Tautologies

$$A \vee B \vee C$$

Is a tautology

- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the clause contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set *S*, it is added to *S*, and is considered for further resolution inferences.

Source: https://en.wikipedia.org/wiki/Resolution_(logic)

# Discarding Tautologies

$$A \vee B \vee C$$

$$\overline{\phantom{B \vee \neg B \vee C \vee D}}$$

$$B \vee \neg B \vee C \vee D$$

Is a tautology

- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the clause contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set $S$, it is added to $S$, and is considered for further resolution inferences.

Source: https://en.wikipedia.org/wiki/Resolution_(logic)

# Discarding Tautologies

$$\cancel{A} \vee B \vee C$$

$$\cancel{\neg A} \vee \neg B \vee D$$

$$B \vee \neg B \vee C \vee D$$

Is a tautology

- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the clause contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set *S*, it is added to *S*, and is considered for further resolution inferences.

Source: https://en.wikipedia.org/wiki/Resolution_(logic)

# Conjunctive Normal Form

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, $\neg$A

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, $\neg$A
  - A and $\neg$A are called complementary literals

- A clause is a disjunction of literals

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols ∨

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, $\neg$A
  - A and $\neg$A are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols $\vee$
  - E.g. A $\vee$ $\neg$B $\vee$ C

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols ∨
  - E.g. $A \vee \neg B \vee C$

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols ∨
  - E.g. A ∨ ¬B ∨ C

- A Knowledge Base is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, $\neg A$
  - A and $\neg A$ are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols $\vee$
  - E.g. $A \vee \neg B \vee C$

- A Knowledge Base is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses
  - That is, clauses joined by conjunction symbols $\wedge$

# Conjunctive Normal Form

- A literal is either a propositional symbol or its negation
  - E.g. A, B, C, ¬A
  - A and ¬A are called complementary literals

- A clause is a disjunction of literals
  - That is, literals joined by disjunction symbols ∨
  - E.g. A ∨ ¬B ∨ C

- A Knowledge Base is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses
  - That is, clauses joined by conjunction symbols ^
  - E.g. (A ∨ ¬B ∨ C) ^ (¬A ∨ C) ^ B

# Converting to CNF

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF
- The following procedure is used:

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
  1. Eliminate $\leftrightarrow$ using the equivalence relationship
     $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
  1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
  2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \land (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \lor b$
    3. Move $\neg$ inward using the following equivalences:

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship
       $a \leftrightarrow b \equiv (a \rightarrow b) \land (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \lor b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF
- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$
        b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship
       $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$
        b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$
        c. $\neg(a \wedge b) \equiv \neg a \vee \neg b$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship
       $a \leftrightarrow b \equiv (a \rightarrow b) \land (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \lor b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$
        b. $\neg(a \lor b) \equiv \neg a \land \neg b$
        c. $\neg(a \land b) \equiv \neg a \lor \neg b$
    4. Distribute $\land$ over $\lor$ using $a \lor (b \land c) \equiv (a \lor b) \land (a \lor c)$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
  1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
  2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
  3. Move $\neg$ inward using the following equivalences:
     a. $\neg\neg a \equiv a$
     b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$
     c. $\neg(a \wedge b) \equiv \neg a \vee \neg b$
  4. Distribute $\wedge$ over $\vee$ using $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
  5. Simplify terms:

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$
        b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$
        c. $\neg(a \wedge b) \equiv \neg a \vee \neg b$
    4. Distribute $\wedge$ over $\vee$ using $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
    5. Simplify terms:
        a. $a \vee a \equiv a$

# Converting to CNF

- A set of sentences in propositional logic can always be converted to CNF

- The following procedure is used:
    1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
    2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
    3. Move $\neg$ inward using the following equivalences:
        a. $\neg\neg a \equiv a$
        b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$
        c. $\neg(a \wedge b) \equiv \neg a \vee \neg b$
    4. Distribute $\wedge$ over $\vee$ using $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
    5. Simplify terms:
        a. $a \vee a \equiv a$
        b. $a \wedge a \equiv a$

# Exercise

- **Convert to CNF** $a \leftrightarrow (b \vee c)$
- The following procedure is used:
  1. Eliminate $\leftrightarrow$ using the equivalence relationship $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
  2. Eliminate $\rightarrow$ using the equivalence relationship $a \rightarrow b \equiv \neg a \vee b$
  3. Move $\neg$ inward using the following equivalences:
     a. $\neg\neg a \equiv a$
     b. $\neg(a \vee b) \equiv \neg a \wedge \neg b$
     c. $\neg(a \wedge b) \equiv \neg a \vee \neg b$
  4. Distribute $\wedge$ over $\vee$ using $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
  5. Simplify terms:
     a. $a \vee a \equiv a$
     b. $a \wedge a \equiv a$

# Alternative notation for clauses

# Alternative notation for clauses

- It is also common to represent clauses as a list
- [¬ A, B, C] is the same as (¬ A v B v C )
- A contradiction is denoted as [ ] (sometimes alto denoted ⊥)

# Proof by resolution

# Proof by resolution

- Input: a set of sentences KB and a query $\alpha$

# Proof by resolution

- Input: a set of sentences KB and a query $\alpha$
- Output: *true* if KB $\models \alpha$, *false* otherwise

# Proof by resolution

- Input: a set of sentences KB and a query $\alpha$
- Output: *true* if KB $\models \alpha$, *false* otherwise
- Strategy: assume KB and $\neg\alpha$ are both true, try to derive a contradiction

# Proof by resolution

- Input: a set of sentences KB and a query $\alpha$
- Output: *true* if KB $\models \alpha$, *false* otherwise
- Strategy: assume KB and $\neg\alpha$ are both true, try to derive a contradiction

# Proof by resolution

# Proof by resolution

**Pseudocode:**

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:
   a. Select two clauses to resolve

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:
   a. Select two clauses to resolve
   b. Add result to KB, unless it is a tautology

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:
   a. Select two clauses to resolve
   b. Add result to KB, unless it is a tautology
   c. If [] is in KB, return *true*

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:
   a. Select two clauses to resolve
   b. Add result to KB, unless it is a tautology
   c. If [] is in KB, return *true*
3. If there are no clauses in KB that resolve to a new clause, return *false*

# Proof by resolution

**Pseudocode:**

1. Add $\neg\alpha$ to the KB
2. While there are clauses in the KB that resolve to a new clause:
   a. Select two clauses to resolve
   b. Add result to KB, unless it is a tautology
   c. If [] is in KB, return *true*
3. If there are no clauses in KB that resolve to a new clause, return *false*

# Resolution example

- $KB = (A \Leftrightarrow (B \vee C)) \wedge \neg A$
- $\alpha = \neg B$
- Intuiton: A is true if and only if B or C holds. But we know A is false. Therefore, B (and also C) must be false)

# Resolution example

- $KB = (A \Leftrightarrow (B \lor C)) \land \neg A$
- $\alpha = \neg B$
- Converting to CNF (see solution to previous exercise), KB =
    1. [¬ A , B , C]
    2. [¬B , A]
    3. [¬C , A]
    4. [¬ A]

# Resolution example

- $KB = (A \Leftrightarrow (B \vee C)) \wedge \neg A$
- $\alpha = \neg B$
- Converting to CNF (see solution to previous exercise), KB =
  1. [¬ A , B , C]
  2. [¬B , A]
  3. [¬C , A]
  4. [¬ A]

# Resolution example

- $KB = (A \Leftrightarrow (B \vee C)) \wedge \neg A$
- $\alpha = \neg B$
- Adding $\neg \alpha = \neg \neg B = B$ to the KB:
    1. $[\neg A, B, C]$
    2. $[\neg B, A]$
    3. $[\neg C, A]$
    4. $[\neg A]$
    5. $[B]$

# Resolution example

# Resolution example

- Applying resolution to pairs of clauses and adding results to KB until we get a contradiction []

1. [¬ A , B , C]
2. [¬B , A]
3. [¬C , A]
4. [¬ A]

# Resolution example

- Applying resolution to pairs of clauses and adding results to KB until we get a contradiction []

1. [¬ A , B , C]
2. [¬B , A]
3. [¬C , A]
4. [¬ A]
5. [B] (negated query)

# Resolution example

- Applying resolution to pairs of clauses and adding results to KB until we get a contradiction []

1. [¬ A , B , C]
2. [¬B , A]
3. [¬C , A]
4. [¬ A]
5. [B] (negated query)
6. [A] (2 , 5)

# Resolution example

- Applying resolution to pairs of clauses and adding results to KB until we get a contradiction []

1. [¬ A , B , C]
2. [¬B , A]
3. [¬C , A]
4. [¬ A]
5. [B] (negated query)
6. [A] (2 , 5)
7. [] (4 , 6)

# First Order Logic - Some equivalences

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
    - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$
- Implication: $\alpha \rightarrow \beta$ is the same as $\neg\alpha \vee \beta$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$
- Implication: $\alpha \rightarrow \beta$ is the same as $\neg\alpha \vee \beta$
  - Sometimes written as $\alpha \implies \beta$ or $\alpha \supset \beta$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$
- Implication: $\alpha \rightarrow \beta$ is the same as $\neg\alpha \vee \beta$
  - Sometimes written as $\alpha \implies \beta$ or $\alpha \supset \beta$
  - $\alpha \rightarrow \beta$ can be understood as "if $\alpha$ is true, I am claiming $\beta$, otherwise, I am making no claim (and hence my claim is true by default)

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$
- Implication: $\alpha \rightarrow \beta$ is the same as $\neg\alpha \vee \beta$
  - Sometimes written as $\alpha \implies \beta$ or $\alpha \supset \beta$
  - $\alpha \rightarrow \beta$ can be understood as "if $\alpha$ is true, I am claiming $\beta$, otherwise, I am making no claim (and hence my claim is true by default)
- Biconditional or equivalence: $\alpha \equiv \beta$ is the same as $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

# First Order Logic - Some equivalences

- Double negation: $\neg\neg\alpha$ is the same as $\alpha$
- de Morgan's laws:
  - $\neg(\alpha \vee \beta)$ is the same as $(\neg\alpha \wedge \neg\beta)$
  - $\neg(\alpha \wedge \beta)$ is the same as $(\neg\alpha \vee \neg\beta)$
- Implication: $\alpha \rightarrow \beta$ is the same as $\neg\alpha \vee \beta$
  - Sometimes written as $\alpha \implies \beta$ or $\alpha \supset \beta$
  - $\alpha \rightarrow \beta$ can be understood as "if $\alpha$ is true, I am claiming $\beta$, otherwise, I am making no claim (and hence my claim is true by default)
- Biconditional or equivalence: $\alpha \equiv \beta$ is the same as $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
  - Sometimes written as $\alpha \leftrightarrow \beta$ $\alpha \iff \beta$ or $\alpha \iff \beta$

# Link between inference and entailment

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
- Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
    - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
    - $S \models \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \models \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \models \bot$ :

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \vDash \bot$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$ and if

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \vDash \perp$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$ and if
  - Resolution is *complete:* if $S \vDash \alpha$ then $S \vdash_R \alpha$ (conversely, if $S \nvdash_R \alpha$ then $S \nvDash \alpha$)

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \models \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \models \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \land \neg\alpha \models \bot$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \models \alpha$ and if
  - Resolution is *complete:* if $S \models \alpha$ then $S \vdash_R \alpha$ (conversely, if $S \nvdash_R \alpha$ then $S \not\models \alpha$)
- For full First-Order Logic, however:

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \vDash \bot$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$ and if
  - Resolution is *complete:* if $S \vDash \alpha$ then $S \vdash_R \alpha$ (conversely, if $S \nvdash_R \alpha$ then $S \nvDash \alpha$)
- For full First-Order Logic, however:
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \vDash \bot$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$ and if
  - Resolution is *complete:* if $S \vDash \alpha$ then $S \vdash_R \alpha$ (conversely, if $S \nvdash_R \alpha$ then $S \nvDash \alpha$)
- For full First-Order Logic, however:
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$
  - But it is not *complete:* there may be cases where $S \nvDash \alpha$ but resolution will never terminate

# Link between inference and entailment

- If an inference procedure $I$ allows us to derive the symbol $\alpha$ from the symbols of the sentences $S$ , we write:
  - $S \vdash_I \alpha$
- If $S$ entails $\alpha$ (that is, every interpretation that satisfies $S$ also satisfies $\alpha$), we write:
  - $S \vDash \alpha$
  - Ideally, we would like a procedure $I$ such that these ideas are the same: $S \vdash_R \alpha$ if and only if $S \vDash \alpha$
- For Propositional logic, this is achieved via Resolution ($R$), typically by contradiction: trying to prove that $KB \wedge \neg\alpha \vDash \perp$ :
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$ and if
  - Resolution is *complete:* if $S \vDash \alpha$ then $S \vdash_R \alpha$ (conversely, if $S \nvdash_R \alpha$ then $S \nvDash \alpha$)
- For full First-Order Logic, however:
  - Resolution is *sound:* if $S \vdash_R \alpha$ then $S \vDash \alpha$
  - But it is *not complete:* there may be cases where $S \nvDash \alpha$ but resolution will never terminate
  - It is however *refutation complete:* if $S \vDash \alpha$ (that is if $S \cup \neg\alpha$ is unsatisfiable), it will terminate and $S \vdash_R \alpha$

# Resolution in First-Order Logic

# Handling variables and quantifiers

# Handling variables and quantifiers

- Strategy: we will again convert the KB to a normal form
  - Similar to CNF, but in addition every variable is universally quantified
  - Note that for any formula $\alpha$, it is true that $\alpha \equiv \forall x \,.\, \alpha$
  - We can then drop all quantifiers
  - Finally, we handle predicates with unification, similar to Prolog
    - Example: P (x,b) unifies with P(a, y) under x/a, y/b

# Handling variables and quantifiers

# Handling variables and quantifiers

- Additional steps in converting to CNF:
  - Move $\neg$ inward of quantifiers
    - $\neg \forall x . \alpha \equiv \exists x . \neg \alpha$
    - $\neg \exists x . \alpha \equiv \forall x . \neg \alpha$
  - Eliminate all other existential quantifiers (see Skolemization at the end)
  - Move $\land$ and $\lor$ inside universal quantifiers
    - $\alpha \land \forall x . \beta \equiv \forall x . \alpha \land \beta$
    - $\alpha \lor \forall x . \beta \equiv \forall x . \alpha \lor \beta$

# Handling variables and quantifiers

# Handling variables and quantifiers

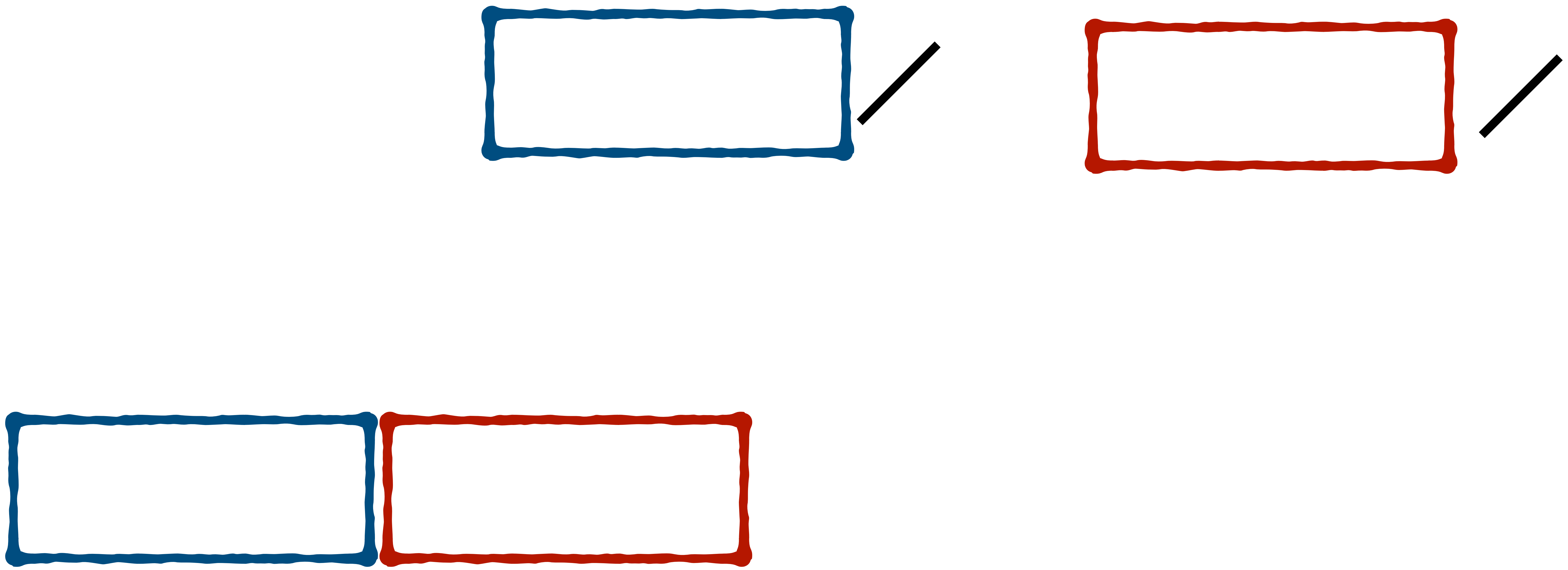
- Assuming no existential quantifiers for now:
- Intuition: if two clauses have $p$ and $\neg q$, but $p$ unifies with $q$, we can apply resolution after performing the required substitution to both clauses

# Handling variables and quantifiers

# Handling variables and quantifiers

- Assuming no existential quantifiers for now:
- Given two clauses $[a_1, a_2, \ldots, a_n, p]$ and $[b_1, b_2, \ldots, b_m, \neg q]$
  - These can be complex clauses with literals containing predicates, variables etc
- If $p$ unifies with $q$ under some substition $\theta$ we get a new clause:
  - $[c_1, c_2, \ldots, c_n, d_1, d2,...,d_m]$
  - Where $c_i = a_i/\theta$ and $d_j = b_j/\theta$

# Handling variables and quantifiers



# Handling variables and quantifiers

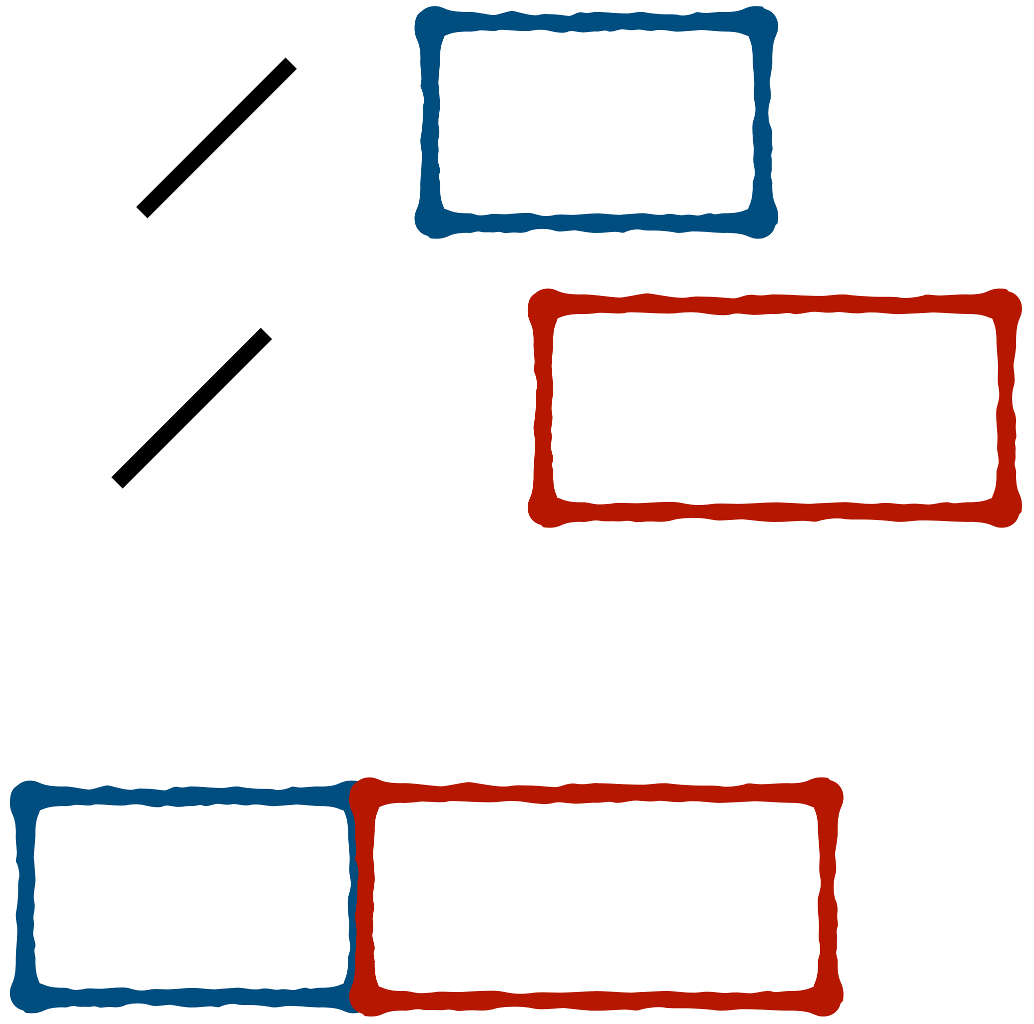
- Assuming no existential quantifiers for now:
- Given two clauses $[\boxed{a_1, a_2, ..., a_n}, \cancel{p}]$ and $[\boxed{b_1, b_2, ..., b_m}, \cancel{\neg q}]$
  - These can be complex clauses with literals containing predicates, variables etc
- If $p$ unifies with $q$ under some substitution $\theta$ we get a new clause:
  - $[\boxed{c_1, c_2, ..., c_n} \boxed{d_1, d2,...,d_m]}$
  - Where $c_i = a_i/\theta$ and $d_j = b_j/\theta$

# Handling variables and quantifiers

# Handling variables and quantifiers

- Example: Consider the clauses:
    - [P(x,a),¬Q(x)]
    - [¬P(b,y), R(b,f(y))]
- The first literal of each clause unifies under x/b, y/a. Applying this to both clauses:
    - [P(b,a),¬Q(b)]
    - [¬P(b,a), R(b,f(a))]
- So we can apply Resolution to eliminate the first literal and get:
    - [¬Q(b), R(b,f(a))]

# Handling variables and quantifiers



# Handling variables and quantifiers

- Example: Consider the clauses:
    - $[P(x,a), \neg Q(x)]$
    - $[\neg P(b,y), R(b,f(y))]$
- The first literal of each clause unifies under x/b, y/a. Applying this to both clauses:
    - $[\cancel{P(b,a)}, \boxed{\neg Q(b)}]$
    - $[\cancel{\neg P(b,a)}, \boxed{R(b,f(a))}]$
- So we can apply Resolution to eliminate the first literal and get:
    - $[\boxed{\neg Q(b)}, \boxed{R(b,f(a))}]$

- $loves(x, mary) \lor happy(mary)$ (implicitly, x is universally quantified)
- $\neg loves(jon, mary)$
- Substitute $x/jon$
- $loves(jon, mary) \lor happy(mary)$
- $\neg loves(jon, mary)$
- We're left with $happy(mary)$

# Skolemization

# Skolemization

- We can replace existential quantifiers that cannot be eliminated by moving ¬ inwards with “dummy” variables and functions called *Skolem constants* and *Skolem* functions

# Skolemization

- We can replace existential quantifiers that cannot be eliminated by moving ¬ inwards with “dummy” variables and functions called *Skolem constants* and *Skolem* functions

- Example: “there exists someone who loves Jane” is replaced by “$anon_1$ loves Jane”

# Skolemization

- We can replace existential quantifiers that cannot be eliminated by moving $\neg$ inwards with "dummy" variables and functions called *Skolem constants* and *Skolem* functions

- Example: "there exists someone who loves Jane" is replaced by "$anon_1$ loves Jane"

    - $\exists x \, . \, loves(x, jane)$ becomes $loves(anon_1, jane)$

# Skolemization

# Skolemization

- Nested quantifiers are a bit more complex:

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z \,.\, friends(x, z) \wedge friends(y, z)$

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z \,.\, friends(x, z) \wedge friends(y, z)$
  - Note that $z$ can depend on $y$, so we can't simply replace $x/anon_1$ and $z/anon_2$

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z \,.\, friends(x, z) \wedge friends(y, z)$
  - Note that $z$ can depend on $y$, so we can't simply replace $x/anon_1$ and $z/anon_2$
    - Let's say Alice is the person with all the common friends

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z \,.\, friends(x, z) \wedge friends(y, z)$
  - Note that $z$ can depend on $y$, so we can't simply replace $x/anon_1$and $z/anon_2$
    - Let's say Alice is the person with all the common friends
    - Bob is the common friend between Alice and Charlie, Dave between Alice and Eleanor…

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z . friends(x, z) \wedge friends(y, z)$
  - Note that $z$ can depend on $y$, so we can't simply replace $x/anon_1$ and $z/anon_2$
    - Let's say Alice is the person with all the common friends
    - Bob is the common friend between Alice and Charlie, Dave between Alice and Eleanor…
  - We introduce a "dummy function" $f(y)$ that returns the common friend for a given $y$

# Skolemization

- Nested quantifiers are a bit more complex:
  - Example: there exists someone who has a common friend with everyone
  - More formally: there exists someone $x$ such that for every person $y$, there exists a third person $z$ that is friends to both
  - $\exists x \forall y \exists z \,.\, friends(x, z) \wedge friends(y, z)$
  - Note that $z$ can depend on $y$, so we can't simply replace $x / anon_1$and $z / anon_2$
    - Let's say Alice is the person with all the common friends
    - Bob is the common friend between Alice and Charlie, Dave between Alice and Eleanor…
  - We introduce a "dummy function" $f(y)$ that returns the common friend for a given $y$
  - The sentence then becomes $\forall y \,.\, friends(anon_1, f(y)) \wedge friends(y, f(y))$

# Conclusion

# Conclusion

- Resolution is a simple but powerful inference procedure

# Conclusion

- Resolution is a simple but powerful inference procedure
- In the propositional case, it is both *sound* and *complete*

# Conclusion

- Resolution is a simple but powerful inference procedure
- In the propositional case, it is both *sound* and *complete*
- In full First-Order Logic, it is *sound* but not *complete*, only *refutation complete*

# Conclusion

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
  - Clauses are disjunctions of literals, KB consists of conjunctions of clauses

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
  - Clauses are disjunctions of literals, KB consists of conjunctions of clauses
  - In full FOL, we also eliminate all existential quantifiers and make every variable universally quantifiable

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
  - Clauses are disjunctions of literals, KB consists of conjunctions of clauses
  - In full FOL, we also eliminate all existential quantifiers and make every variable universally quantifiable
- We then try to resolve pairs of clauses until we either get an empty clause (meaning $KB \models \alpha$) or we fail to produce new clauses (meaning $KB \not\models \alpha$)

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
  - Clauses are disjunctions of literals, KB consists of conjunctions of clauses
  - In full FOL, we also eliminate all existential quantifiers and make every variable universally quantifiable
- We then try to resolve pairs of clauses until we either get an empty clause (meaning $KB \models \alpha$) or we fail to produce new clauses (meaning $KB \not\models \alpha$)
- In full FOL, we may need to *unify* terms before resolving clauses

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
  - Clauses are disjunctions of literals, KB consists of conjunctions of clauses
  - In full FOL, we also eliminate all existential quantifiers and make every variable universally quantifiable
- We then try to resolve pairs of clauses until we either get an empty clause (meaning $KB \vDash \alpha$) or we fail to produce new clauses (meaning $KB \nvDash \alpha$)
  - In full FOL, we may need to *unify* terms before resolving clauses
- In both propositional and FOL, inference can take an exponential number of steps

# Conclusion

- To apply resolution, we first convert the KB and negated query to CNF
    - Clauses are disjunctions of literals, KB consists of conjunctions of clauses
    - In full FOL, we also eliminate all existential quantifiers and make every variable universally quantifiable
- We then try to resolve pairs  of clauses until we either get an empty clause (meaning $KB \models \alpha$) or we fail to produce new clauses (meaning $KB \not\models \alpha$)
    - In full FOL, we may need to *unify* terms before resolving clauses
- In both propositional and FOL, inference can take an exponential number of steps
    - Faster procedures are needed

# Next Chapter

# Next Chapter

- We will look at inference with *Horn clauses*

# Next Chapter

- We will look at inference with *Horn clauses*
- Prolog is built around Horn clauses!

# Next Chapter

- We will look at inference with *Horn clauses*
- Prolog is built around Horn clauses!
- Less expressive than full FOL, but resolution, if it terminates, uses only a linear number of steps

# Next Chapter

- We will look at inference with *Horn clauses*
- Prolog is built around Horn clauses!
- Less expressive than full FOL, but resolution, if it terminates, uses only a linear number of steps
- Unfortunately, the problem is still undecidable: no sound procedure can be guaranteed to terminate

# Exercise: who killed the cat?

(from Russel & Norvig's “AI - a Modern Approach”, ch. 9)

# Exercise

# Exercise

Given the following English sentences:

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

E. Did Curiosity kill the cat?

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

E. Did Curiosity kill the cat?

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

E. Did Curiosity kill the cat?

**Task 1:** write these as First-Order Logic sentences

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

E. Did Curiosity kill the cat?

**Task 1:** write these as First-Order Logic sentences

**Task 2:** convert each sentence to CNF

# Exercise

Given the following English sentences:

A. Everyone who loves all animals is loved by someone

B. Anyone who kills an animal is loved by no one

C. Jack loves all animals

D. Either Jack or Curiosity killed the cat, who is named Tuna

E. Did Curiosity kill the cat?

**Task 1:** write these as First-Order Logic sentences

**Task 2:** convert each sentence to CNF

**Task 3:** write an informal proof that Curiosity killed the cat (in English)

# Exercise

Given the following English sentences:

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

**Task 1:** write these as First-Order Logic sentences

**Task 2:** convert each sentence to CNF

**Task 3:** write an informal proof that Curiosity killed the cat (in English)

**Task 4:** Use resolution to answer the question “Did Curiosity kill the cat?”

# Task 1 - Converting to CNF

# Task 1 - Converting to CNF

Given the following English sentences:

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x (\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x(\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x[\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x (\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x [\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x(\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x[\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$Did Curiosity kill the cat?

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x (\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x [\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$Did Curiosity kill the cat?

D. 2) Cat(Tuna)

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x(\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x[\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$Did Curiosity kill the cat?

D. 2) Cat(Tuna)

D. 3) $\forall_x cat(x) \rightarrow animal(x)$

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x (\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x [\exists_z animal(z) \land kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$Did Curiosity kill the cat?

D. 2) Cat(Tuna)

D. 3) $\forall_x cat(x) \rightarrow animal(x)$

E. $\neg kills(Curiosity, Tuna)$ (negated query)

# Task 1 - Converting to CNF

Given the following English sentences:

A. $\forall_x (\forall_y animal(y) \rightarrow loves(x, y)) \rightarrow \exists_z loves(z, x)$

B. $\forall_x [\exists_z animal(z) \wedge kills(x, z)] \rightarrow \forall_y \neg loves(y, x)$

C. $\forall_x animal(x) \rightarrow loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$Did Curiosity kill the cat?

D. 2) Cat(Tuna)

D. 3) $\forall_x cat(x) \rightarrow animal(x)$

E. $\neg kills(Curiosity, Tuna)$ (negated query)

# Task 2 - From English to FOL

# Task 2 - From English to FOL

Given the following English sentences:

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \vee loves(G(x), x)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \vee loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \vee loves(G(x), x)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \vee loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \vee loves(G(x), x)$

B. $\neg animal(z) \vee \neg kills(x, z) \vee \neg loves(y, x)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \lor loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \lor loves(G(x), x)$

B. $\neg animal(z) \lor \neg kills(x, z) \lor \neg loves(y, x)$

C. $\neg animal(x) \lor loves(Jack, x)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \lor loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \lor loves(G(x), x)$

B. $\neg animal(z) \lor \neg kills(x, z) \lor \neg loves(y, x)$

C. $\neg animal(x) \lor loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \vee loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \vee loves(G(x), x)$

B. $\neg animal(z) \vee \neg kills(x, z) \vee \neg loves(y, x)$

C. $\neg animal(x) \vee loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$

D. 2) Cat(Tuna)

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \lor loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \lor loves(G(x), x)$

B. $\neg animal(z) \lor \neg kills(x, z) \lor \neg loves(y, x)$

C. $\neg animal(x) \lor loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$

D. 2) Cat(Tuna)

D. 3) $\neg cat(x) \lor animal(x)$

# Task 2 - From English to FOL

Given the following English sentences:

A. 1) $animal(F(x)) \vee loves(G(x), x)$

A. 2) $\neg loves(x, F(x)) \vee loves(G(x), x)$

B. $\neg animal(z) \vee \neg kills(x, z) \vee \neg loves(y, x)$

C. $\neg animal(x) \vee loves(Jack, x)$

D. 1) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$

D. 2) Cat(Tuna)

D. 3) $\neg cat(x) \vee animal(x)$

E. $\neg kills(Curiosity, Tuna)$

# Task 2 - From English to FOL (renumbering and renaming variables)

# Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

## Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \vee loves(G(x_1), x_1)$

# Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \vee loves(G(x_1), x_1)$

2. $\neg loves(x_2, F(x_2)) \vee loves(G(x_2), x_2)$

# Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \vee loves(G(x_1), x_1)$

2. $\neg loves(x_2, F(x_2)) \vee loves(G(x_2), x_2)$

3. $\neg animal(z_3) \vee \neg kills(x_3, z_3) \vee \neg loves(y_3, x_3)$

## Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \vee loves(G(x_1), x_1)$
2. $\neg loves(x_2, F(x_2)) \vee loves(G(x_2), x_2)$
3. $\neg animal(z_3) \vee \neg kills(x_3, z_3) \vee \neg loves(y_3, x_3)$
4. $\neg animal(x_4) \vee loves(Jack, x_4)$

## Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \lor loves(G(x_1), x_1)$

2. $\neg loves(x_2, F(x_2)) \lor loves(G(x_2), x_2)$

3. $\neg animal(z_3) \lor \neg kills(x_3, z_3) \lor \neg loves(y_3, x_3)$

4. $\neg animal(x_4) \lor loves(Jack, x_4)$

5. $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$

# Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \lor loves(G(x_1), x_1)$
2. $\neg loves(x_2, F(x_2)) \lor loves(G(x_2), x_2)$
3. $\neg animal(z_3) \lor \neg kills(x_3, z_3) \lor \neg loves(y_3, x_3)$
4. $\neg animal(x_4) \lor loves(Jack, x_4)$
5. $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$
6. Cat(Tuna)

# Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \vee loves(G(x_1), x_1)$
2. $\neg loves(x_2, F(x_2)) \vee loves(G(x_2), x_2)$
3. $\neg animal(z_3) \vee \neg kills(x_3, z_3) \vee \neg loves(y_3, x_3)$
4. $\neg animal(x_4) \vee loves(Jack, x_4)$
5. $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$
6. Cat(Tuna)
7. $\neg cat(x_7) \vee animal(x_7)$

## Task 2 - From English to FOL (renumbering and renaming variables)

Given the following English sentences:

1. $animal(F(x_1)) \lor loves(G(x_1), x_1)$
2. $\neg loves(x_2, F(x_2)) \lor loves(G(x_2), x_2)$
3. $\neg animal(z_3) \lor \neg kills(x_3, z_3) \lor \neg loves(y_3, x_3)$
4. $\neg animal(x_4) \lor loves(Jack, x_4)$
5. $kills(Jack, Tuna) \lor kills(Curiosity, Tuna)$
6. Cat(Tuna)
7. $\neg cat(x_7) \lor animal(x_7)$
8. $\neg kills(Curiosity, Tuna)$

# Task 3 - Proof by resolution

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

# Task 3 - Proof by resolution

9. $animal(Tuna)$

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

# Task 3 - Proof by resolution

9. $animal(Tuna)$

10. $\neg kills(x, Tuna) \lor \neg loves(y, x)$

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

# Task 3 - Proof by resolution

9. $animal(Tuna)$

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$

11. $kills(Jack, Tuna)$

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

# Task 3 - Proof by resolution

9. $animal(Tuna)$ (6,7, x/Tuna)

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ (3,9, z/Tuna)

11. $kills(Jack, Tuna)$ (5,8)

12. $\neg loves(y, Jack)$ (10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

# Task 3 - Proof by resolution

| | |
|---|---|
| 9. $animal(Tuna)$ | (6,7, x/Tuna) |
| 10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ | (3,9, z/Tuna) |
| 11. $kills(Jack, Tuna)$ | (5,8) |
| 12. $\neg loves(y, Jack)$ | (10,11, x/Jack) |
| 13. $\neg animal(F(Jack)) \vee loves(G(Jack), Jack)$ | (2,4, $x_2/Jack, x_4/F(Jack)$) |
| | (1,13, $x_1/Jack$ ) |
| | (12,14, $y/G(Jack)$ ) |

# Task 3 - Proof by resolution

| | |
|---|---|
| 9. $animal(Tuna)$ | (6,7, x/Tuna) |
| 10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ | (3,9, z/Tuna) |
| 11. $kills(Jack, Tuna)$ | (5,8) |
| 12. $\neg loves(y, Jack)$ | (10,11, x/Jack) |
| 13. $\neg animal(F(Jack)) \vee loves(G(Jack), Jack)$ | (2,4, $x_2/Jack, x_4/F(Jack)$) |
| 14. $loves(G(Jack), Jack)$ | (1,13, $x_1/Jack$ ) |
| | (12,14, $y/G(Jack)$ ) |

# Task 3 - Proof by resolution

| | |
|---|---|
| 9. $animal(Tuna)$ | (6,7, x/Tuna) |
| 10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ | (3,9, z/Tuna) |
| 11. $kills(Jack, Tuna)$ | (5,8) |
| 12. $\neg loves(y, Jack)$ | (10,11, x/Jack) |
| 13. $\neg animal(F(Jack)) \vee loves(G(Jack), Jack)$ | (2,4, $x_2/Jack, x_4/F(Jack)$) |
| 14. $loves(G(Jack), Jack)$ | (1,13, $x_1/Jack$ ) |
| 15. $\bot$ | (12,14, $y/G(Jack)$ ) |

# Task 3 - Proof by resolution

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

9. $animal(Tuna)$

(6,7, x/Tuna)

(3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack$, $x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

9. $animal(Tuna)$ (6,7, x/Tuna)

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ (3,9, z/Tuna)

(5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

9. $animal(Tuna)$ (6,7, x/Tuna)

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ (3,9, z/Tuna)

11. $kills(Jack, Tuna)$ (5,8)

(10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

9. $animal(Tuna)$ (6,7, x/Tuna)

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ (3,9, z/Tuna)

11. $kills(Jack, Tuna)$ (5,8)

12. $\neg loves(y, Jack)$ (10,11, x/Jack)

(2,4, $x_2/Jack, x_4/F(Jack)$)

(1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

| | |
|---|---|
| 9. $animal(Tuna)$ | (6,7, x/Tuna) |
| 10. $\neg kills(x, Tuna) \lor \neg loves(y, x)$ | (3,9, z/Tuna) |
| 11. $kills(Jack, Tuna)$ | (5,8) |
| 12. $\neg loves(y, Jack)$ | (10,11, x/Jack) |
| 13. $\neg animal(F(Jack)) \lor loves(G(Jack), Jack)$ | (2,4, $x_2/Jack, x_4/F(Jack)$) |
| | (1,13, $x_1/Jack$ ) |
| | (12,14, $y/G(Jack)$ ) |

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

9. $animal(Tuna)$ — (6,7, x/Tuna)

10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ — (3,9, z/Tuna)

11. $kills(Jack, Tuna)$ — (5,8)

12. $\neg loves(y, Jack)$ — (10,11, x/Jack)

13. $\neg animal(F(Jack)) \vee loves(G(Jack), Jack)$ — (2,4, $x_2/Jack, x_4/F(Jack)$)

14. $loves(G(Jack), Jack)$ — (1,13, $x_1/Jack$ )

(12,14, $y/G(Jack)$ )

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution

| | |
|---|---|
| 9. $animal(Tuna)$ | (6,7, x/Tuna) |
| 10. $\neg kills(x, Tuna) \vee \neg loves(y, x)$ | (3,9, z/Tuna) |
| 11. $kills(Jack, Tuna)$ | (5,8) |
| 12. $\neg loves(y, Jack)$ | (10,11, x/Jack) |
| 13. $\neg animal(F(Jack)) \vee loves(G(Jack), Jack)$ | (2,4, $x_2/Jack, x_4/F(Jack)$) |
| 14. $loves(G(Jack), Jack)$ | (1,13, $x_1/Jack$ ) |
| 15. $\perp$ | (12,14, $y/G(Jack)$ ) |

Note:step 13 requires us to rename the variables (x appears on both sides)

# Task 3 - Proof by resolution (alternative visualization from AIMA)

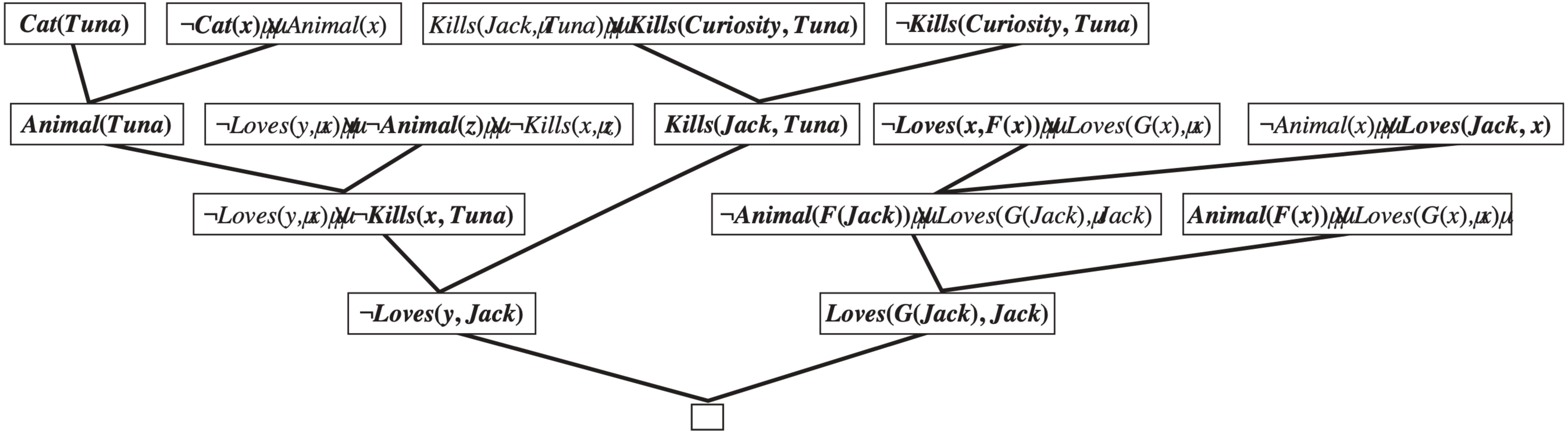



**Figure 9.12** A resolution proof that Curiosity killed the cat. Notice the use of factoring in the derivation of the clause $Loves(G(Jack), Jack)$. Notice also in the upper right, the unification of $Loves(x, F(x))$ and $Loves(Jack, x)$ can only succeed after the variables have been standardized apart.