Propositional Logic

21 October 2020
You can think about deep learning as equivalent to ... our visual cortex or auditory cortex. But, of course, true intelligence is a lot more than just that, you have to recombine it into higher-level thinking and symbolic reasoning, a lot of the things classical AI tried to deal with in the 80s... We would like to build up to this symbolic level of reasoning – maths, language, and logic. So that’s a big part of our work.

Demis Hassabis, CEO of Google Deepmind
Knowledge representation
We can consider a declarative approach to building an agent (or other system):

*Tell* it what it needs to know (or have it *Learn* the knowledge)

Then it can *Ask* itself what to do – answers should follow from the KB
We can represent knowledge about the world as a knowledge base consisting of sentences – facts and rules – expressed in some formal knowledge representation language.
An agent can use an *inference procedure* to reason using that knowledge, often by asking a question.

A single inference algorithm can answer *any* answerable question.

This contrasts with a search algorithm, which answers only “how to get from A to B” questions.
The kinds of knowledge represented, and the form of the reasoning are heavily dependent on the knowledge representation language we use.
**Logic**

**Syntax:** What sentences are allowed?

**Semantics:**

What are the *possible* worlds?

Which sentences are true in which worlds? (i.e., *definition* of truth)

\[ a_1, a_2, a_3 \]

Syntaxland

Semanticsland
Logic

Syntax: What sentences are allowed?

Semantics:

What are the possible worlds?

Which sentences are true in which worlds? (i.e., definition of truth)

Syntaxland

Semanticsland
Logic

Syntax: What sentences are allowed?

Semantics:

What are the possible worlds?

Which sentences are true in which worlds? (i.e., definition of truth)

\[ a_1 \]

\[ a_2 \quad a_3 \]

Syntaxland

Semanticsland
Logic

**Syntax:** What sentences are allowed?

**Semantics:**

What are the *possible* worlds?

Which sentences are true in which worlds? (i.e., *definition* of truth)
Logic

Syntax: What sentences are allowed?

Semantics:

What are the possible worlds?

Which sentences are true in which worlds? (i.e., definition of truth)

$\alpha_1$

$\alpha_2$

$\alpha_3$
Logic

**Syntax**: What sentences are allowed?

**Semantics**: What are the *possible* worlds?
Which sentences are true in which worlds? (i.e., *definition* of truth)
Propositional logic
A *proposition* is a statement that is, by itself, either *true* or *false*.

*Propositional logic* is a formal knowledge representation language and a set of inference rules for reasoning about propositions.
Every statement in propositional logic consists of *propositional variables*.

Each variable represents some proposition, e.g.,

“I am near a velociraptor” or “I will be eaten by a velociraptor”.

Using *propositional connectives*, we can build new propositions out of existing ones.

Connectives encode how propositions are related, e.g.,

“If I am near a velociraptor, then I will be eaten by a velociraptor.”
Because words are ambiguous, we use symbols to represent the propositional connectives, just as we do with numerical operations like +, −, or <.
Logical negation

\[ \neg \varphi \]

Read as “not \( \varphi \)”. 
Logical conjunction

\( \varphi \land \psi \)

Read as “\( \varphi \) and \( \psi \)”. 
Logical disjunction

\[ \varphi \lor \psi \]

Read as “\( \varphi \text{ or } \psi \)”. 
Inclusive vs exclusive disjunction

In words, “φ or ψ” could mean:

**Exclusive:**
- Host: “You may have coffee or tea.”
- Guest: “Coffee, please.”

**Inclusive:**
- Host: “Would you like cream or sugar?”
- Guest: “Both, please.”
Inclusive vs exclusive disjunction

The $\lor$ connective is *inclusive or* – it’s true if at least one of the operands is true.

This matches the behavior of `or` in Python.

If we want *exclusive or* – where exactly one operand is true – we can build it out of the connectives we already have.
Logical implication

\[ \varphi \implies \psi \]

Read as “\( \varphi \) implies \( \psi \)”. 
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

$p \quad q \quad p \Rightarrow q$

Today the role of “copper ingots” will be played by these bricks.
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

Nanni pays Ea-Nasir

Ea-Nasir gives quality ingots

Contract upheld?

Nanni

Ea-Nasir
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

Nanni pays Ea-Nasir

Ea-Nasir gives quality ingots

Contract upheld?

\[ p \quad q \quad p \Rightarrow q \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Nanni

Ea-Nasir
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

Nanni **pays** Ea-Nasir

Ea-Nasir gives **quality** ingots

Contract upheld?

Nanni: $\text{T}$

Ea-Nasir: $\text{T}$

$p \rightarrow q$ holds true for all possible scenarios.
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

Nanni pays Ea-Nasir

Ea-Nasir gives quality ingots

Contract upheld?

\[ p \quad \Rightarrow \quad q \]

<table>
<thead>
<tr>
<th>Nanni</th>
<th>F</th>
<th>T</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
</table>

| Ea-Nasir | F | F | T | T |
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.

Nanni **pays**

Ea-Nasir

Ea-Nasir gives **quality** ingots

Contract upheld?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Nanni

Ea-Nasir
Ancient Babylonian contract

If Nanni pays money to Ea-Nasir, then Ea-Nasir will give Nanni high-quality copper ingots.
An implication is false only when the antecedent is true and the consequent is false.

Every formula is either true or false, so these other entries need to be true.
Observe that $\varphi \Rightarrow \psi$ is true whenever $\varphi \land \neg \psi$ is false.

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\psi$</th>
<th>$\varphi \Rightarrow \psi$</th>
<th>$\varphi \land \neg \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Observe that $\varphi \Rightarrow \psi$ is true whenever $\varphi \land \neg \psi$ is false.

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\psi$</th>
<th>$\varphi \Rightarrow \psi$</th>
<th>$\varphi \land \neg (\varphi \land \neg \psi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>$\neg (\varphi \land \neg \psi)$</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>$\neg (\varphi \land \neg \psi)$</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>$\neg (\varphi \land \neg \psi)$</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>$\neg (\varphi \land \neg \psi)$</td>
</tr>
</tbody>
</table>
An implication with a false antecedent is called vacuously true.
Biconditional implication

\[ \phi \iff \psi \]

Read as “\( \phi \) if and only if \( \psi \)”. 
One interpretation of $\leftrightarrow$ is to think of it as equality: The two propositions must have equal truth values.

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\psi$</th>
<th>$\varphi \leftrightarrow \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
One interpretation of $\iff$ is to think of it as equality: The two propositions must have equal truth values.

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\psi$</th>
<th>$\varphi \iff \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>$\neq$</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>$\neq$</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>$\neq$</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
True and false

In propositional logic, we can also use True and False directly.

We can write out the words or we can use the symbol \(\top\) for true and \(\bot\) for false.

These are often called connectives, like \(\land\), \(\lor\), \(\neg\), \(\Rightarrow\), and \(\Leftrightarrow\), but they connect zero things.
Operator precedence

\( \neg x \Rightarrow y \lor z \Rightarrow x \lor y \land z \)

How do we parse this statement?
Operator precedence

\( \neg x \Rightarrow y \lor z \Rightarrow x \lor y \land z \)

How do we parse this statement?

Precedence: \( \neg \land \lor \Rightarrow \leftrightarrow \)

We can use parentheses to disambiguate.
Operator precedence

\( \neg x \Rightarrow y \lor z \Rightarrow x \lor y \land z \)

How do we parse this statement?

Precedence: \( \neg \land \lor \Rightarrow \leftrightarrow \)

We can use parentheses to disambiguate.
Operator precedence

\[(\neg x) \Rightarrow y \lor z \Rightarrow x \lor y \land z\]

How do we parse this statement?

Precedence: \[\neg \land \lor \Rightarrow \leftrightarrow\]

We can use parentheses to disambiguate.
Operator precedence

$$(\neg x) \Rightarrow y \lor z \Rightarrow x \lor y \land z$$

How do we parse this statement?

<table>
<thead>
<tr>
<th>Precedence: $\neg \land \lor \Rightarrow \leftrightarrow$</th>
</tr>
</thead>
</table>

We can use parentheses to disambiguate.
Operator precedence

\((\neg x) \Rightarrow y \lor z \Rightarrow x \lor (y \land z)\)

How do we parse this statement?

Precendence: \(\neg \land \lor \Rightarrow \leftrightarrow\)

We can use parentheses to disambiguate.
Operator precedence

$$(\neg x) \Rightarrow y \lor z \Rightarrow x \lor (y \land z)$$

How do we parse this statement?

Precedence: $\neg \land \lor \Rightarrow \iff$

We can use parentheses to disambiguate.
Operator precedence

\[ (\neg x) \Rightarrow (y \lor z) \Rightarrow (x \lor (y \land z)) \]

How do we parse this statement?

Precedence: \( \neg \land \lor \Rightarrow \leftrightarrow \)

We can use parentheses to disambiguate.
Operator precedence

\[(\neg x) \Rightarrow (y \lor z) \Rightarrow (x \lor (y \land z))\]

How do we parse this statement?

Precedence: \(\neg \land \lor \Rightarrow \leftrightarrow\)

We can use parentheses to disambiguate.
Operator precedence

$$(\neg x) \Rightarrow (((y \lor z) \Rightarrow (x \lor (y \land z))))$$

How do we parse this statement?

Precedence: $\neg \land \lor \Rightarrow \leftrightarrow$

We can use parentheses to disambiguate.
Operator precedence

Logical operators have an order of operations, just like mathematical operators.

From low to high: negation, conjunction, disjunction, implication

You can think of conjunction like multiplication and disjunction like addition

**Math:** \(-k \cdot (x + y)\)

**Logic:** \(\neg p \land (q \lor r)\)
The **syntax** of a representation language tells us how we can write a sentence.

In *arithmetic*,

\[ x + y = 4 \] is a well-formed sentence.

\[ x 4 y + \] is not.

In *propositional logic*,

\[ p \lor \neg (q \Rightarrow r) \] is a well-formed sentence.

\[ \land p \neg q r \] is not.
The **semantics** of a formal representation tells us whether each sentence is true, with respect to some possible world.

In *arithmetic*, \( x + y = 4 \)

- is true in a world where \( x \) is 2 and \( y \) is 2
- is false in a world where \( x \) is 1 and \( y \) is 1

In *propositional logic*, \( p \lor q \)

- is true in a world where \( p \) is true and \( q \) is false
- is false in a world where \( p \) is false and \( q \) is false
A propositional knowledge base is a list of sentences that apply to the world, e.g.,

\[
\begin{align*}
\text{Cold} \\
\neg\text{Raining} \\
\text{Raining} \lor \text{Cloudy} \\
\text{Cold} \leftrightarrow \neg\text{Hot}
\end{align*}
\]

A knowledge base characterizes the set of \textit{possible worlds} in which these facts and rules are true.
A *model* is a formalization of a “world”.

Every proposition symbol in the KB is set to *True* or *False*.

If there are $n$ propositions, there are $2^n$ models possible.

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Value</th>
<th>Proposition</th>
<th>Value</th>
<th>Proposition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>False</td>
<td>Cold</td>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raining</td>
<td>False</td>
<td>Raining</td>
<td>False</td>
<td>Raining</td>
<td>True</td>
</tr>
<tr>
<td>Cloudy</td>
<td>False</td>
<td>Cloudy</td>
<td>False</td>
<td>Cloudy</td>
<td>True</td>
</tr>
<tr>
<td>Hot</td>
<td>False</td>
<td>Hot</td>
<td>False</td>
<td>Hot</td>
<td>True</td>
</tr>
</tbody>
</table>

...
Each sentence has a truth value in each model.

If a sentence \( \alpha \) is true in model \( M \), then \( M \) satisfies \( \alpha \).

A KB specifies a subset of all possible models – those that satisfy all sentences in the KB.

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>False</td>
</tr>
<tr>
<td>Raining</td>
<td>False</td>
</tr>
<tr>
<td>Cloudy</td>
<td>False</td>
</tr>
<tr>
<td>Hot</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>True</td>
</tr>
<tr>
<td>Raining</td>
<td>False</td>
</tr>
<tr>
<td>Cloudy</td>
<td>False</td>
</tr>
<tr>
<td>Hot</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>True</td>
</tr>
<tr>
<td>Raining</td>
<td>True</td>
</tr>
<tr>
<td>Cloudy</td>
<td>True</td>
</tr>
<tr>
<td>Hot</td>
<td>True</td>
</tr>
</tbody>
</table>

\[ \neg \text{Raining} \]

\[ \text{Raining} \lor \text{Cloudy} \]

\[ \text{Cold} \leftrightarrow \neg \text{Hot} \]
For a proposition, or a KB of propositions, if it’s true in at least one model, it is *satisfiable*. It isn’t true in any model is called *unsatisfiable*.

Unsatisfiable propositions are also known as *contradictions*. 
A prisoner was told that by making a statement, he could choose the method of his execution; if the statement was true, he would be shot, and if false, he would be hanged.

The prisoner made the statement, “I shall be hanged”.
Propositional logic semantics

def pl_true?(α, model):
    if α is a symbol:
        return lookup(α, model)
    if α.operator = ¬:
        return not pl_true?(α.arg1, model)
    if α.operator = ∧:
        return (pl_true?(α.arg1, model) and pl_true?(α.arg2, model))
    if α.operator = ⇒:
        return (pl_true?(α.arg1, model) or not pl_true?(α.arg2, model))
Summary

Knowledge base

Set of facts and rules asserted to be true about the world

Model

Formalization of “the world”
An assignment of values to all variables

Satisfaction

A model satisfies a sentence if that sentence is true in the model.
A model satisfies a KB if all of the sentences are true in the model.
Reasoning
So, if we have a KB, then what?

Given:

Cold
¬Raining
Raining ∨ Cloudy
Cold ↔ ¬Hot

We’d like to ask questions, e.g.,

Is it Hot?
**Inference** is the process of deriving new facts from what’s known.

Desirable properties:

- Don’t make any mistakes.
- Be able to prove all possible true statements.
**Entailment**: $\alpha \models \beta$ ("$\alpha$ entails $\beta$" or "$\beta$ follows from $\alpha$") if and only if, in every world where $\alpha$ is true, $\beta$ is also true.

I.e., $\text{models}(\alpha) \subseteq \text{models}(\beta)$
**Entailment:** $a \models \beta$ (“$a$ entails $\beta$” or “$\beta$ follows from $a$”) if and only if, in every world where $a$ is true, $\beta$ is also true.

I.e., $\text{models}(a) \subseteq \text{models}(\beta)$
Entailment: \(a \vDash \beta\) ("\(a\) entails \(\beta\)" or "\(\beta\) follows from \(a\)") if and only if, in every world where \(a\) is true, \(\beta\) is also true.

I.e., \(\text{models}(a) \subseteq \text{models}(\beta)\)
**Entailment:** $\alpha \models \beta$ ("$\alpha$ entails $\beta$" or "$\beta$ follows from $\alpha$") if and only if, in every world where $\alpha$ is true, $\beta$ is also true.

I.e., $\text{models}(\alpha) \subseteq \text{models}(\beta)$
Entailment: $\alpha \models \beta$ (“$\alpha$ entails $\beta$” or “$\beta$ follows from $\alpha$”) if and only if, in every world where $\alpha$ is true, $\beta$ is also true.

I.e., $\text{models}(\alpha) \subseteq \text{models}(\beta)$

In the example, $\alpha_2 \models \alpha_1$

(E.g., $\alpha_2$ might be $\neg Q \land R \land S \land W$ and $\alpha_1$ be $\neg Q$)
The knowledge base $A$ entails sentence $B$ ($A \models B$) if and only if every model that satisfies $A$ also satisfies $B$.

“Don’t make any mistakes.”

An inference method is sound if it only produces propositions that are actually entailed by the KB.

“Be able to prove all possible true statements.”

An inference method is complete if it can produce every proposition that’s entailed by the KB.
How should we do this?

Two general approaches:

*Model-checking*

We can enumerate models (possible worlds) and check if the claim is true in every model of the KB.

This can work for propositional logic, where there are finitely many worlds, but it’s not easy for first-order logic!

*Theorem-proving*

Search for a sequence of applications of *inference rules*, e.g.,

\[
A \lor B, \neg B
\]

\[
\frac{A \lor B, \neg B}{A}
\]
We want to start somewhere (KB).
We'd like to apply some rules.
There are lots of ways we might go.
We want to reach some goal (sentence).
Sound familiar?
Inference as search

*Set of states*: True sentences

*Start state*: KB

*Set of actions and action rules*: Inference rules

*Goal test*: \( Q \) in sentences?

*Cost function*: 1 per rule
Pac-Man facts

If Pac-Man is at [3, 3] at time 16 and goes North and there is no wall at [3, 4], then Pac-Man is at [3, 4] at time 17:

\[ At_{3,3}^{16} \land N^{16} \land \neg Wall_{3,4} \Rightarrow At_{3,3}^{17} \]
Pac-Man facts

If Pac-Man is at [3, 3] at time 16 and goes North and there is no wall at [3, 4], then Pac-Man is at [3, 4] at time 17:

\[ At_{3,3}^{16} \land N^{16} \land \neg Wall_{3,4} \Rightarrow At_{3,3}^{17} \]

At time 0, Pac-Man does one of four actions:

\[ (W^0 \lor E^0 \lor N^0 \lor S^0) \]
Pac-Man facts

If Pac-Man is at [3, 3] at time 16 and goes North and there is no wall at [3, 4], then Pac-Man is at [3, 4] at time 17:

\[ \text{At}_{3,3}^{16} \land N^{16} \land \neg \text{Wall}_{3,4} \Rightarrow \text{At}_{3,3}^{17} \]

At time 0, Pac-Man does one of four actions:

\[ (W^0 \lor E^0 \lor N^0 \lor S^0) \]

\[ \neg (W^0 \land E^0) \land \neg (W^0 \land S^0) \land \cdots \]
Observation $\rightarrow$ Inference (syntactic) $\rightarrow$ True in the world (semantics)
Inference: Theorem-proving
A familiar example of a sound inference rule is *modus ponens*:

\[
\phi \Rightarrow \psi, \phi \\
\hline
\psi
\]

or

\[
\{\phi \Rightarrow \psi, \phi\} \models \psi
\]
Simple theorem proving: Forward chaining

Forward chaining applies *modus ponens* to generate new facts:

Given $X_1 \land X_2 \land \ldots \land X_n \Rightarrow Y$ and $X_1, X_2, \ldots, X_n$

Infer $Y$

Forward chaining keeps applying this rule, adding new facts, until nothing more can be added.
Simple theorem proving: Forward chaining

Forward chaining applies *modus ponens* to generate new facts:

Given $X_1 \land X_2 \land \ldots \land X_n \Rightarrow Y$ and $X_1, X_2, \ldots, X_n$

Infer $Y$

Forward chaining keeps applying this rule, adding new facts, until nothing more can be added.

Requires KB to contain only *definite clauses:*

(Conjunction of symbols) $\Rightarrow$ symbol; or

A single symbol. (Note that $X$ is equivalent to $\text{True} \Rightarrow X$.)
Forward chaining algorithm

```python
def pl_fc_entails?(KB, q):
    count = a table,
    where count[c] is the number of symbols in c’s premise
    inferred = a table,
    where inferred[s] is initially false for all s
    agenda = a queue of symbols,
    initially symbols known to be true in KB
    while agenda is not empty:
        p = Pop(agenda)
        if p == q:
            return True
        if inferred[p] == False:
            inferred[p] = True
            for each clause c in KB where p is in c.premise:
                count[c] -= 1
                if count[c] == 0:
                    add c.conclusion to agenda
    return False
```
## Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>A \ False</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>B \ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>L \ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>2</td>
<td>M \ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>2</td>
<td>P \ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>Q \ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>2</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>2</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

A  B
Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>1</td>
<td>A False</td>
</tr>
<tr>
<td>( L \land M \Rightarrow P )</td>
<td>2</td>
<td>B False</td>
</tr>
<tr>
<td>( B \land L \Rightarrow M )</td>
<td>2</td>
<td>L False</td>
</tr>
<tr>
<td>( A \land P \Rightarrow L )</td>
<td>2</td>
<td>M False</td>
</tr>
<tr>
<td>( A \land B \Rightarrow L )</td>
<td>2</td>
<td>P False</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>Q False</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda
A  B
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>2</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>2</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

- $\times B$
**Forward chaining example: Proving Q**

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P \Rightarrow Q)</td>
<td>1</td>
<td>(A\ False\ True)</td>
</tr>
<tr>
<td>(L \land M \Rightarrow P)</td>
<td>2</td>
<td>(B\ False)</td>
</tr>
<tr>
<td>(B \land L \Rightarrow M)</td>
<td>2</td>
<td>(L\ False)</td>
</tr>
<tr>
<td>(A \land P \Rightarrow L)</td>
<td>2</td>
<td>(M\ False)</td>
</tr>
<tr>
<td>(A \land B \Rightarrow L)</td>
<td>2</td>
<td>(P\ False)</td>
</tr>
<tr>
<td>(A)</td>
<td>0</td>
<td>(Q\ False)</td>
</tr>
<tr>
<td>(B)</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

- \(\times B\)
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False  True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times$ $B$
Forward chaining example: Proving \( Q \)

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>1</td>
<td>( \text{False} \text{ True} )</td>
</tr>
<tr>
<td>( L \land M \Rightarrow P )</td>
<td>2</td>
<td>( \text{False} )</td>
</tr>
<tr>
<td>( B \land L \Rightarrow M )</td>
<td>2</td>
<td>( \text{False} )</td>
</tr>
<tr>
<td>( A \land P \Rightarrow L )</td>
<td>1</td>
<td>( \text{False} )</td>
</tr>
<tr>
<td>( A \land B \Rightarrow L )</td>
<td>1</td>
<td>( \text{False} )</td>
</tr>
<tr>
<td>( A )</td>
<td>0</td>
<td>( \text{False} )</td>
</tr>
<tr>
<td>( B )</td>
<td>0</td>
<td>( \text{False} )</td>
</tr>
</tbody>
</table>

Agenda
- \( \times \times \times \)
# Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False, True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False, True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>✗ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>✗ 1</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

✗ ✗ ✗
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $;\Box$ $\text{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False $;\Box$ $\text{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>√ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>√ 1</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda:

$\Box \Box \Box$
# Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $\Rightarrow$ $True$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False $\Rightarrow$ $True$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times$ 1</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times$ $\times$ 0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

$\times$ $\times$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$\text{False}$ $\text{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$\text{False}$ $\text{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>1</td>
<td>$\text{False}$</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>1</td>
<td>$\text{False}$</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td>$\text{False}$</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$\text{False}$</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td>$\text{False}$</td>
</tr>
</tbody>
</table>

Agenda

$\text{False}$ $\text{True}$$\text{False}$ $\text{False}$ $\text{False}$ $\text{False}$
### Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $\textbf{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False $\textbf{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times$ 1</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

$\times \times L$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $True$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False $True$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times$ 1</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times$ $\times$ 0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times$ $\times$ $L$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td>False</td>
</tr>
</tbody>
</table>

Agenda

$x 
$x 
$x 
$x 

Diagram of the proof.
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Inferred:
- $A$: False, True
- $B$: False, True
- $L$: False, True
- $M$: False
- $P$: False
- $Q$: False

Agenda:
- $\times \times \times \times$
### Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False, $B$ True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>2</td>
<td>$B$ False, $L$ True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>2</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>2</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>2</td>
<td>$Q$ False</td>
</tr>
</tbody>
</table>

**Agenda**

- $\times \times \times $
### Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $\quad$ True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times$ 1</td>
<td>$B$ False $\quad$ True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times$ 0</td>
<td>$L$ False $\quad$ True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

$\times \times \times$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>A False  True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>![x] 1</td>
<td>B False  True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>![x] ![x] 0</td>
<td>L False  True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>![x] 1</td>
<td>M False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>![x] ![x] 0</td>
<td>P False</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>Q False</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

![x] ![x] ![x]
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>0</td>
<td>$B$ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>0</td>
<td>$L$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>0</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
</tbody>
</table>

Agenda:

$M$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$\text{False}$ $\text{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times$ 1</td>
<td>$\text{False}$ $\text{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times$ 0</td>
<td>$\text{False}$ $\text{True}$</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ $\text{False}$</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$P$ $\text{False}$</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ $\text{False}$</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

$\times \times \times \times \times M$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $\text{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times$ 1</td>
<td>$B$ False $\text{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times$ 0</td>
<td>$L$ False $\text{True}$</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times$ 1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times \times$
## Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( L \land M \Rightarrow P )</td>
<td>1</td>
<td>( A ) False True</td>
</tr>
<tr>
<td>( B \land L \Rightarrow M )</td>
<td>0</td>
<td>( B ) False True</td>
</tr>
<tr>
<td>( A \land P \Rightarrow L )</td>
<td>0</td>
<td>( L ) False True</td>
</tr>
<tr>
<td>( A \land B \Rightarrow L )</td>
<td>0</td>
<td>( M ) False True</td>
</tr>
<tr>
<td>( A )</td>
<td>0</td>
<td>( P ) False</td>
</tr>
<tr>
<td>( B )</td>
<td>0</td>
<td>( Q ) False</td>
</tr>
</tbody>
</table>

### Agenda

\( \times \times \times \times \times \)
## Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False  $B$ False  $P$ False  $Q$ False</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>1</td>
<td>$B$ False  $M$ False</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>0</td>
<td>$L$ False  $M$ False</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>1</td>
<td>$M$ False</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Agenda

$x\times x\times x\times x$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times \times 0$</td>
<td>$B$ False True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times 0$</td>
<td>$L$ False True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times 1$</td>
<td>$M$ False True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times 0$</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>× × 0</td>
<td>$A$ False True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>× × 0</td>
<td>$B$ False True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>× 1</td>
<td>$L$ False True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>× × 0</td>
<td>$M$ False True</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

× × × ×
## Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>A False  True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>✗ ✗ 0</td>
<td>B False  True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>✗ ✗ 0</td>
<td>L False  True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>✗ 1</td>
<td>M False  True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>✗ ✗ 0</td>
<td>P False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>Q False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

* ✗ ✗ ✗ ✗ ✗ $P$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>$A$ False $\Rightarrow$ True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times \times 0$</td>
<td>$B$ False $\Rightarrow$ True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times 0$</td>
<td>$L$ False $\Rightarrow$ True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times 1$</td>
<td>$M$ False $\Rightarrow$ True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times 0$</td>
<td>$P$ False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times \times P$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>1</td>
<td>A False  True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>× × 0</td>
<td>B False  True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>× × 0</td>
<td>L False  True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>× 1</td>
<td>M False  True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>× × 0</td>
<td>P False</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>Q False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

× × × × ×
### Forward chaining example: Proving \( Q \)

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \rightarrow Q )</td>
<td>1</td>
<td>( A ) True</td>
</tr>
<tr>
<td>( L \land M \rightarrow P )</td>
<td>2 * 2</td>
<td>( B ) False True</td>
</tr>
<tr>
<td>( B \land L \rightarrow M )</td>
<td>2 * 2</td>
<td>( L ) False True</td>
</tr>
<tr>
<td>( A \land P \rightarrow L )</td>
<td>2 * 2</td>
<td>( M ) False True</td>
</tr>
<tr>
<td>( A \land B \rightarrow L )</td>
<td>2 * 2</td>
<td>( P ) False True</td>
</tr>
<tr>
<td>( A )</td>
<td>0</td>
<td>( Q ) False</td>
</tr>
<tr>
<td>( B )</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

\* \* \* \* \* \*
### Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q$</td>
<td>1</td>
<td>$A$ False $True$</td>
</tr>
<tr>
<td>$L \land M \rightarrow P$</td>
<td>$\times \times 0$</td>
<td>$B$ False $True$</td>
</tr>
<tr>
<td>$B \land L \rightarrow M$</td>
<td>$\times \times 0$</td>
<td>$L$ False $True$</td>
</tr>
<tr>
<td>$A \land P \rightarrow L$</td>
<td>$\times 1$</td>
<td>$M$ False $True$</td>
</tr>
<tr>
<td>$A \land B \rightarrow L$</td>
<td>$\times \times 0$</td>
<td>$P$ False $True$</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

$\times \times \times \times \times$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>$\times 0$</td>
<td>$A$ False $\mathbf{True}$</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times \times 0$</td>
<td>$B$ False $\mathbf{True}$</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times 0$</td>
<td>$L$ False $\mathbf{True}$</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times \times 0$</td>
<td>$M$ False $\mathbf{True}$</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times 0$</td>
<td>$P$ False $\mathbf{True}$</td>
</tr>
<tr>
<td>$A$</td>
<td>$0$</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>$0$</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times \times$
**Forward chaining example: Proving Q**

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>( \times ) 0</td>
<td>( A ) False True</td>
</tr>
<tr>
<td>( L \land M \Rightarrow P )</td>
<td>( \times \times ) 0</td>
<td>( B ) False True</td>
</tr>
<tr>
<td>( B \land L \Rightarrow M )</td>
<td>( \times \times ) 0</td>
<td>( L ) False True</td>
</tr>
<tr>
<td>( A \land P \Rightarrow L )</td>
<td>( \times \times ) 0</td>
<td>( M ) False True</td>
</tr>
<tr>
<td>( A \land B \Rightarrow L )</td>
<td>( \times \times ) 0</td>
<td>( P ) False True</td>
</tr>
<tr>
<td>( A )</td>
<td>0</td>
<td>( Q ) False</td>
</tr>
<tr>
<td>( B )</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

\( \times \times \times \times \times \)
### Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>☒ 0</td>
<td></td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>☒ ☒ 0</td>
<td></td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>☒ ☒ 0</td>
<td></td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>☒ ☒ 0</td>
<td></td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>☒ ☒ 0</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒</td>
</tr>
</tbody>
</table>
Forward chaining example: Proving $Q$

### Clauses

<table>
<thead>
<tr>
<th>Clause</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>$\times$ 0</td>
<td>$A$ True $B$ True $L$ True $M$ True $P$ True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>$\times \times$ 0</td>
<td>$B$ True $L$ True $M$ True $P$ True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>$\times \times$ 0</td>
<td>$L$ True $M$ True $P$ True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$M$ True $P$ True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>$\times \times$ 0</td>
<td>$P$ True</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Agenda

$\times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \time...
Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$L \land M \rightarrow P$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$B \land L \rightarrow M$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$A \land P \rightarrow L$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$A \land B \rightarrow L$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>$0$</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>$0$</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\bigstar \bigstar \bigstar \bigstar \bigstar \bigstar L Q$
Forward chaining example: Proving Q

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>× 0</td>
<td>A False  True</td>
</tr>
<tr>
<td>( L \land M \Rightarrow P )</td>
<td>× × 0</td>
<td>B False  True</td>
</tr>
<tr>
<td>( B \land L \Rightarrow M )</td>
<td>× × 0</td>
<td>L False  True</td>
</tr>
<tr>
<td>( A \land P \Rightarrow L )</td>
<td>× × 0</td>
<td>M False  True</td>
</tr>
<tr>
<td>( A \land B \Rightarrow L )</td>
<td>× × 0</td>
<td>P False  True</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>Q False</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda:

× × × × × × × Q
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>× 0</td>
<td></td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>× × 0</td>
<td></td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>× × 0</td>
<td></td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>× × 0</td>
<td></td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>× × 0</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$P$</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>$L$</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>$M$</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>$Q$</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

× × × × × × × × $Q$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \Rightarrow Q$</td>
<td>0</td>
<td>$A$ False True</td>
</tr>
<tr>
<td>$L \land M \Rightarrow P$</td>
<td>0</td>
<td>$B$ False True</td>
</tr>
<tr>
<td>$B \land L \Rightarrow M$</td>
<td>0</td>
<td>$L$ False True</td>
</tr>
<tr>
<td>$A \land P \Rightarrow L$</td>
<td>0</td>
<td>$M$ False True</td>
</tr>
<tr>
<td>$A \land B \Rightarrow L$</td>
<td>0</td>
<td>$P$ False True</td>
</tr>
<tr>
<td>$A$</td>
<td>0</td>
<td>$Q$ False</td>
</tr>
<tr>
<td>$B$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times \times \times \times$
Forward chaining example: Proving $Q$

<table>
<thead>
<tr>
<th>Clauses</th>
<th>Count</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$L \land M \rightarrow P$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$B \land L \rightarrow M$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$A \land P \rightarrow L$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$A \land B \rightarrow L$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$A$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
<tr>
<td>$B$</td>
<td>$0$</td>
<td>$\text{False} \rightarrow \text{True}$</td>
</tr>
</tbody>
</table>

Agenda

$\times \times \times \times \times \times \times \times$
Forward chaining is sound and complete for definite-clause KBs.
Resolution

This inference rule is both sound and complete when it’s combined with a sound and complete search algorithm:

\[
\begin{align*}
    a_1 \lor \cdots \lor a_{i-1} \lor \cdots \lor a_n, \quad b_1 \lor \cdots \lor b_{j-1} \lor \cdots \lor b_m \\
    \hline
    a_1 \lor \cdots \lor a_{i-1} \lor a_{i+1} \lor \cdots \lor a_n \lor b_1 \lor \cdots \lor b_{j-1} \lor \cdots \lor b_m
\end{align*}
\]

But to use it, the inputs need to be in conjunctive normal form.
Conjunctive normal form (CNF)

Every sentence can be expressed as a *conjunction of clauses*.

Each *clause* is a *disjunction of literals*.

Each *literal* is a symbol or a negated symbol.

Conversion to CNF can be done using a sequence of standard transformations.
\[ \text{At}_{1,1}^0 \Rightarrow (\text{Wall}_{0,1} \iff \text{Blocked}_{W}^0) \]

\[ \text{At}_{1,1}^0 \Rightarrow ((\text{Wall}_{0,1} \Rightarrow \text{Blocked}_{W}^0) \land (\text{Blocked}_{W}^0 \Rightarrow \text{Wall}_{0,1})) \]

\[ \neg \text{At}_{1,1}^0 \lor ((\neg \text{Wall}_{0,1} \lor \text{Blocked}_{W}^0) \land (\neg \text{Blocked}_{W}^0 \lor \text{Wall}_{0,1})) \]

\[ (\neg \text{At}_{1,1}^0 \lor \neg \text{Wall}_{0,1} \lor \text{Blocked}_{W}^0) \land (\neg \text{At}_{1,1}^0 \lor \neg \text{Blocked}_{W}^0 \lor \text{Wall}_{0,1}) \]

Replace biconditional by two implications

Replace \( a \Rightarrow \beta \) by \( \neg a \lor \beta \)

Distribute \( \lor \) over \( \land \)
Inference: Model-checking
Simple model checking

```python
def tt_entails?(KB, α):
    return tt_check_all(KB, α, symbols(KB) ∪ symbols(α), {})

def tt_check_all(KB, α, symbols, model):
    if empty?(symbols):
        if pl_true?(KB, model):
            return pl_true?(α, model)
        else:
            return True
    else:
        P = first(symbols)
        return(tt_check_all(KB, α, rest(symbols),
                           model ∪ {P = True}) and
                tt_check_all(KB, α, rest(symbols),
                           model ∪ {P = False}))
```
Simple model checking, contd.
Simple model checking, contd.

\[ P_1 = \text{true} \]
\[ P_2 = \text{true} \]
\[ P_n = \text{true} \]

\[ P_1 = \text{false} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{false} \]
Simple model checking, contd.

\[ P_1 = \text{true} \]
\[ P_1 = \text{false} \]
\[ P_2 = \text{true} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{true} \]
\[ P_n = \text{false} \]
Simple model checking, contd.

Same recursion as backtracking

$$P_1 = true$$

$$P_1 = false$$

$$P_2 = true$$

$$P_2 = false$$

$$P_n = true$$

$$P_n = false$$

11111...1

0000...0
Simple model checking, contd.

Same recursion as backtracking

\[ P_1 = \text{true} \]
\[ P_1 = \text{false} \]
\[ P_2 = \text{true} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{false} \]
\[ P_n = \text{true} \]

KB?
Simple model checking, contd.

Same recursion as backtracking

$P_1 = \text{true}$
$P_2 = \text{true}$
$P_n = \text{true}$

$P_1 = \text{false}$
$P_2 = \text{false}$
$P_n = \text{false}$

$KB?$

11111...1

0000...0
Simple model checking, contd.

Same recursion as backtracking

\[ P_1 = \text{true} \]
\[ P_1 = \text{false} \]
\[ P_2 = \text{true} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{false} \]
\[ P_n = \text{true} \]
Simple model checking, contd.

Same recursion as backtracking

$P_1 = \text{true}$

$P_1 = \text{false}$

$P_2 = \text{true}$

$P_2 = \text{false}$

$P_n = \text{false}$

$P_n = \text{true}$

$\alpha$?

$KB?$

11111...1

0000...0
Simple model checking, contd.

Same recursion as backtracking

$P_1 = \text{true}$

$P_1 = \text{false}$

$P_2 = \text{true}$

$P_2 = \text{false}$

$P_n = \text{false}$

$P_n = \text{true}$

KB? α?

11111\ldots 1

0000\ldots 0
Simple model checking, contd.

Same recursion as backtracking

\begin{align*}
P_1 &= \text{true} \\
P_1 &= \text{false} \\
P_2 &= \text{true} \\
P_2 &= \text{false} \\
P_n &= \text{false} \\
P_n &= \text{true} \\
\end{align*}

KB?

\begin{align*}
\alpha &\,
\end{align*}
Simple model checking, contd.

Same recursion as backtracking

\[ P_1 = \text{true} \]

\[ P_1 = \text{false} \]

\[ P_2 = \text{true} \]

\[ P_2 = \text{false} \]

\[ P_n = \text{false} \]

\[ P_n = \text{true} \]

\[ \text{KB?} \]

\[ a? \]

\[ 11111...1 \]

\[ 0000...0 \]
Simple model checking, contd.

Same recursion as backtracking

$P_1 = \text{true}$

$P_1 = \text{false}$

$P_2 = \text{true}$

$P_2 = \text{false}$

$P_n = \text{false}$

$P_n = \text{true}$

$KB? \ a?$

11111...1

0000...0
Simple model checking, contd.

Same recursion as backtracking

\[ P_1 = \text{true} \]
\[ P_1 = \text{false} \]
\[ P_2 = \text{true} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{false} \]
\[ P_n = \text{true} \]

KB?
a?

11111...1

0000...0
Simple model checking, contd.

Same recursion as backtracking
Simple model checking, contd.

Same recursion as backtracking

$P_1 = \text{true}$
$P_1 = \text{false}$

$P_2 = \text{true}$
$P_2 = \text{false}$

$P_n = \text{false}$
$P_n = \text{true}$

$KB? a?$
Simple model checking, contd.

Same recursion as backtracking
Simple model checking, contd.

Same recursion as backtracking

\[ P_1 = \text{true} \]
\[ P_1 = \text{false} \]
\[ P_2 = \text{true} \]
\[ P_2 = \text{false} \]
\[ P_n = \text{false} \]
\[ P_n = \text{true} \]
Simple model checking, contd.

Same recursion as backtracking

$O(2^n)$ time, linear space
Simple model checking, contd.

Same recursion as backtracking

$O(2^n)$ time, linear space

We can do much better!
Satisfiability and entailment
Satisfiability and entailment

Remember: A sentence is *satisfiable* if it’s true in at least one world (cf. CSPs!)}
Satisfiability and entailment

Remember: A sentence is *satisfiable* if it’s true in at least one world (cf. CSPs!)

Suppose we have a hyper-efficient SAT solver. How could we use it to test entailment?

Suppose \( a \models \beta \)
Then \( a \Rightarrow \beta \) is *true* in all worlds
Hence \( \neg (a \Rightarrow \beta) \) is *false* in all worlds
Hence \( a \land \neg \beta \) is false in all worlds, i.e., unsatisfiable
Satisfiability and entailment

Remember: A sentence is *satisfiable* if it’s true in at least one world (cf. CSPs!)

Suppose we have a hyper-efficient SAT solver. How could we use it to test entailment?

Suppose $\alpha \models \beta$

Then $\alpha \Rightarrow \beta$ is *true* in all worlds

Hence $\neg(\alpha \Rightarrow \beta)$ is *false* in all worlds

Hence $\alpha \land \neg\beta$ is false in all worlds, i.e., unsatisfiable

So, add the negated conclusion to what you know and test for *(un)satisfiability* – also known as *reductio ad absurdum.*
Efficient SAT solvers

**DPLL** (Davis–Putnam–Logemann–Loveland) is the core of modern solvers.

Essentially a backtracking search over models with some extras:

*Early termination*: Stop if

- all clauses are satisfied; e.g., \((A \lor B) \land (A \lor \neg C)\) is satisfied by \(\{A = \text{true}\}\)
- any clause is falsified; e.g., \((A \lor B) \land (A \lor \neg C)\) can’t be satisfied by \(\{A = \text{false}, B = \text{false}\}\)

**Pure literals**: If all occurrences of a symbol in as-yet-unsatisfied clauses have the same sign, then give the symbol that value

E.g., \(A\) is pure and positive in \((A \lor B) \land (A \lor \neg C) \land (C \lor \neg B)\), so set it to true

**Unit clauses**: If a clause is left with a single literal, set symbol to satisfy clause

E.g., if \(A = \text{false}\), \((A \lor B) \land (A \lor \neg C)\) becomes \((\text{false} \lor B) \land (\text{false} \lor \neg C)\), i.e., \((B) \land (\neg C)\)

Satisfying the unit clauses often leads to further propagation, new unit clauses, etc.
DPLL algorithm

```python
def dpll(clauses, symbols, model):
    if every clause in clauses is true in model:
        return True
    if some clause in clauses is false in model:
        return False
    P, value = find_pure_symbol(symbols, clauses, model)
    if P is non-null:
        return dpll(clauses, symbols - P, model \cup \{P=value\})
    P, value = find_unit_clause(clauses, model)
    if P is non-null:
        return dpll(clauses, symbols - P, model \cup \{P=value\})
    P = first(symbols)
    rest = rest(symbols)
    return (dpll(clauses, rest, model \cup \{P=true\}) or
dpll(clauses, rest, model \cup \{P=false\}))```
Efficiency
Efficiency

Naïve implementation of DPLL: solve ~100 variables
Efficiency

Naïve implementation of DPLL: solve ~100 variables

Extras:

- Variable and value ordering (from CSPs)
- Divide and conquer
- Caching unsolvable subcases as extra clauses to avoid redoing them
- Cool indexing and incremental recomputation tricks so that every step of the DPLL algorithm is efficient (typically $O(1)$)
  - Index of clauses in which each variable appears +ve/−ve
  - Keep track number of satisfied clauses, update when variables assigned
  - Keep track of number of remaining literals in each clause
Efficiency

Naïve implementation of DPLL: solve ~100 variables

Extras:

- Variable and value ordering (from CSPs)
- Divide and conquer
- Caching unsolvable subcases as extra clauses to avoid redoing them
- Cool indexing and incremental recomputation tricks so that every step of the DPLL algorithm is efficient (typically O(1))
  - Index of clauses in which each variable appears +ve/-ve
  - Keep track number of satisfied clauses, update when variables assigned
  - Keep track of number of remaining literals in each clause

Real implementation of DPLL: solve ~10 000 000 variables
Knowledge-based agents
SAT solvers in practice
SAT solvers in practice

*Circuit verification*: Does this VLSI circuit compute the right answer?
SAT solvers in practice

*Circuit verification*: Does this VLSI circuit compute the right answer?

*Software verification*: Does this program compute the right answer?
SAT solvers in practice

*Circuit verification*: Does this VLSI circuit compute the right answer?

*Software verification*: Does this program compute the right answer?

*Software synthesis*: What program computes the right answer?
SAT solvers in practice

*Circuit verification:* Does this VLSI circuit compute the right answer?

*Software verification:* Does this program compute the right answer?

*Software synthesis:* What program computes the right answer?

*Protocol verification:* Can this security protocol be broken?
SAT solvers in practice

*Circuit verification*: Does this VLSI circuit compute the right answer?

*Software verification*: Does this program compute the right answer?

*Software synthesis*: What program computes the right answer?

*Protocol verification*: Can this security protocol be broken?

*Protocol synthesis*: What protocol is secure for this task?
SAT solvers in practice

*Circuit verification*: Does this VLSI circuit compute the right answer?

*Software verification*: Does this program compute the right answer?

*Software synthesis*: What program computes the right answer?

*Protocol verification*: Can this security protocol be broken?

*Protocol synthesis*: What protocol is secure for this task?

*Planning*: How can I eat all the dots?
A knowledge-based agent

```python
def kb_agent(kb, percept, t):
    tell(kb, make_percept_sentence(percept, t))
    action = ask(kb, make_action_query(t))
    tell(kb, make_action_sentence(action, t))
    t += 1
    return action
```
Example: Partially observable Pac-Man
Example: Partially observable Pac-Man

Pac-Man has to act given only local perception

Four Boolean percept variables for wall in each direction
Example: Partially observable Pac-Man

Pac-Man has to act given only local perception

Four Boolean percept variables for wall in each direction

What knowledge does he need to begin with?

*Sensor model:* sentences stating how the current percept variables are determined by the current state variables

*Transition model:* sentences stating how the next-state variables are determined by the current state variables and Pac-Man’s action

*Initial conditions:* what Pac-Man knows about the initial state

*Domain constraints:* what is generally true, e.g., Pac-Man can do one thing at a time and be in one place at a time
Pac-Man variables
Pac-Man variables

Pac-Man’s location, e.g., $A_{t,1}^0$: Pac-Man is at [1, 1] at time 0
Pac-Man variables

Pac-Man’s location, e.g., $At_{1,1}^0$: Pac-Man is at $[1, 1]$ at time 0

Wall locations, e.g., $Wall_{0,0}$, $Wall_{0,1}$, etc. (These don’t change with time.)
Pac-Man variables

Pac-Man’s location, e.g., $At_{1,1}^0$: Pac-Man is at $[1, 1]$ at time 0

Wall locations, e.g., $Wall_{0,0}$, $Wall_{0,1}$, etc. (These don’t change with time.)

Percepts, e.g., $Blocked_W^0$: Blocked by wall to my West at time 0.
Pac-Man variables

Pac-Man’s location, e.g., $At_{1,1}^0$: Pac-Man is at [1, 1] at time 0

Wall locations, e.g., $Wall_{0,0}$, $Wall_{0,1}$, etc. (These don’t change with time.)

Percepts, e.g., $Blocked^W_0$: Blocked by wall to my West at time 0.

Actions, e.g, $W^0$: Pac-Man moves West at time 0
Pac-Man variables

Pac-Man’s location, e.g., $At_{1,1}^0$: Pac-Man is at [1, 1] at time 0

Wall locations, e.g., $Wall_{0,0}$, $Wall_{0,1}$, etc. (These don’t change with time.)

Percepts, e.g., $Blocked_W^0$: Blocked by wall to my West at time 0.

Actions, e.g, $W^0$: Pac-Man moves West at time 0

An $N \times N$ world for $T$ time steps leads to $N^2T + N^2 + 4T + 4T = O(N^2T)$ variables.
Pac-Man variables

Pac-Man’s location, e.g., $At_{1,1}^0$: Pac-Man is at $[1, 1]$ at time 0

Wall locations, e.g., $Wall_{0,0}$, $Wall_{0,1}$, etc. (These don’t change with time.)

Percepts, e.g., $Blocked_W^0$: Blocked by wall to my West at time 0.

Actions, e.g, $W^0$: Pac-Man moves West at time 0

An $N \times N$ world for $T$ time steps leads to $N^2T + N^2 + 4T + 4T = O(N^2T)$ variables.

That means $2^{N^2T}$ possible worlds! $N = 10$, $T = 100$ leads to $10^{3010}$ worlds, each a “history”.
Sensor model
Sensor model

State facts about how Pac-Man’s percepts arise…
Sensor model

State facts about how Pac-Man’s percepts arise…

Pac-Man perceives a wall to the West at time $t$ if and only if he is in $[x, y]$ and there is a wall at $[x-1, y]$…

\[ \text{Blocked}_W^0 \iff \]

\[ (((At_{1,1}^0 \land Wall_{0,1}) \lor (At_{1,2}^0 \land Wall_{0,2}) \lor (At_{1,3}^0 \land Wall_{0,3}) \lor \cdots) \]
Sensor model

State facts about how Pac-Man’s percepts arise...

Pac-Man perceives a wall to the West at time $t$ if and only if he is in $[x, y]$ and there is a wall at $[x-1, y]$...

$$\text{Blocked}_W^0 \iff ((\text{At}_{1,1}^0 \land \text{Wall}_{0,1}) \lor (\text{At}_{1,2}^0 \land \text{Wall}_{0,2}) \lor (\text{At}_{1,3}^0 \land \text{Wall}_{0,3}) \lor \cdots)$$

How many of these sentences?
Sensor model

State facts about how Pac-Man’s percepts arise…

Pac-Man perceives a wall to the West at time $t$ if and only if he is in $[x, y]$ and there is a wall at $[x-1, y]$…

$Blocked^0_W \iff$

$((At_{1,1}^0 \land Wall_{0,1}) \lor$

$(At_{1,2}^0 \land Wall_{0,2}) \lor$

$(At_{1,3}^0 \land Wall_{0,3}) \lor \cdots)$

How many of these sentences? How big are they?
Quiz

What’s wrong with sentences like

\[ At_{1,1}^0 \land Wall_{0,1} \Rightarrow Blocked_{W}^0 \]

“If you’re at [1, 1] at time 0 and there’s a wall in [0, 1], the west percept is blocked.”
Quiz

What’s wrong with sentences like

\[ At_{1,1}^0 \land Wall_{0,1} \Rightarrow Blocked_{W}^0 \]

“If you’re at [1, 1] at time 0 and there’s a wall in [0, 1], the west percept is blocked.”

True – but incomplete!

They say “under these conditions the percept variable is true”.

They don’t say when it is false.

In particular, they allow for worlds where the percept is always true!
Transition model
Transition model

How does each *state variable* or *fluent* at each time get its value?
Transition model

How does each state variable or fluent at each time gets its value?

State variables for PO-Pac-Man are $A_{x,y}^t$, e.g., $A_{3,3}^{17}$
Transition model

How does each state variable or fluent at each time gets its value?

State variables for PO-Pac-Man are $A_{t,x,y}^t$, e.g., $A_{3,3}^{17}$

A state variable gets its value according to a successor-state axiom

$$X^t \leftrightarrow [X^{t-1} \land \neg (\text{some action}^{t-1} \text{ made it false})] \lor$$

$$[\neg X^{t-1} \land (\text{some action}^{t-1} \text{ made it true})]$$
Transition model

How does each state variable or fluent at each time get its value?

State variables for PO-Pac-Man are $At_{x,y}^t$, e.g., $At_{3,3}^{17}$

A state variable gets its value according to a successor-state axiom

$$X^t \Leftrightarrow [X^{t-1} \land \neg\text{(some action}^{t-1} \text{ made it false})] \lor$$
$$[\neg X^{t-1} \land \text{(some action}^{t-1} \text{ made it true})]$$

For Pac-Man location:

$$At_{3,3}^{17} \Leftrightarrow [At_{3,3}^{16} \land \neg((\neg Wall_{3,4} \land N^{16}) \lor (\neg Wall_{4,3} \land E^{16}) \lor \cdots)] \lor$$
$$[\neg At_{3,3}^{16} \land ((At_{3,2}^{16} \land \neg Wall_{3,3} \land N^{16}) \lor (At_{2,3}^{16} \land \neg Wall_{3,3} \land N^{16}) \lor \cdots)]$$
Initial state
Initial state

Pac-Man may know its initial location:

$\mathcal{A}_{t_{1,1}}^{0}$
Initial state

Pac-Man may know its initial location:

\[ \text{At}_{1,1}^0 \land \neg \text{At}_{1,2}^0 \land \neg \text{At}_{1,3}^0 \cdots \]
Initial state

Pac-Man may know its initial location:

\[ At_{1,1}^0 \land \neg At_{1,2}^0 \land \neg At_{1,3}^0 \cdots \]

Or, it may not:

\[ At_{1,1}^0 \lor At_{1,2}^0 \lor At_{1,3}^0 \lor \cdots \lor At_{3,3}^0 \]
Initial state

Pac-Man may know its initial location:

\[ \text{At}_{1,1}^0 \land \neg \text{At}_{1,2}^0 \land \neg \text{At}_{1,3}^0 \ldots \]

Or, it may not:

\[ \text{At}_{1,1}^0 \lor \text{At}_{1,2}^0 \lor \text{At}_{1,3}^0 \lor \ldots \lor \text{At}_{3,3}^0 \]

We also need a domain constraint – exactly one thing at a time

\[ \neg (W^0 \land E^0) \land \neg (W^0 \land S^0) \land \ldots \]

\[ \neg (W^1 \land E^1) \land \neg (W^1 \land S^1) \land \ldots \]

\[ \ldots \land (W^0 \lor E^0 \lor N^0 \lor S^0) \land \ldots \]
State estimation means keeping track of what's true now.

A logical agent can just ask itself!

E.g., ask whether $KB \land \langle \text{actions} \rangle \land \langle \text{percepts} \rangle \models \text{Wall}_{2,2}$

This is “lazy”: it involves reasoning about one’s whole life history at each step!
**State estimation** means keeping track of what's true now.

A logical agent can just ask itself!

E.g., ask whether $KB \land \langle actions \rangle \land \langle percepts \rangle \models Wall_{2,2}$

This is “lazy”: it involves reasoning about one’s whole life history at each step!

A more “eager” form of state estimation:

After each action and percept,

For each state variable $X_t$:

If $X_t$ is entailed, add to KB.

If $\neg X_t$ is entailed, add to KB.
Planning as satisfiability

Given a hyper-efficient SAT solver, can we use it to make plans?

Yes, for fully observable, deterministic case:

- Planning problem is solvable iff there is some satisfying assignment
- Solution obtained from truth values of action variables

For $T = 1$ to infinity, set up the KB as follows and run SAT solver:

- Initial state, domain constraints
- Transition model sentences up to time $T$
- Goal is true at time $T$

Read off action variables from solution
Summary

One possible agent architecture: knowledge and inference.

Logics provide a formal way to encode knowledge.

A logic is defined by: syntax, set of possible worlds, and truth condition.

Logical inference computes entailment relations among sentences.

SAT solvers based on DPLL provide incredibly efficient inference.

Logical agents can construct plans by asking whether there is a future in which the goal is achieved.
Appendix: Resolution
Propositional resolution is a rule of inference that is *refutation complete*, i.e., using resolution, we can derive a contradiction whenever a set of propositions – a *knowledge base* – is unsatisfiable.
Using resolution is like doing a proof by contradiction.

To see if a proposition, called the goal, is entailed by a KB, we add its negation to the KB and apply resolution.

If we derive a contradiction, the goal is entailed by the KB.
If we run out of propositions to generate without deriving a contradiction, the goal is not entailed by the KB.
Propositional resolution only works for propositions that are in *conjunctive normal form* (CNF).
This is in CNF 😊

$p \equiv (q \lor (\neg r \Rightarrow s))$

This is not 😞
Every proposition can be converted into an equivalent proposition in CNF:

1. Replace any \( \Rightarrow \) or \( \iff \) by equivalent formulas using \( \land \), \( \lor \), and \( \neg \):
   
   \[
   a \Rightarrow \beta \quad \text{changes to} \quad \neg a \lor \beta
   \]
   
   \[
   a \iff \beta \quad \text{changes to} \quad (a \land \beta) \lor (\neg a \land \neg \beta)
   \]

2. Move negation inward, using de Morgan’s laws and eliminating double negations:
   
   \[
   \neg(a \land \beta) \quad \text{changes to} \quad \neg a \lor \neg \beta
   \]
   
   \[
   \neg(a \lor \beta) \quad \text{changes to} \quad \neg a \land \neg \beta
   \]
   
   \[
   \neg
   \]
   
   \[
   \quad \text{changes to} \quad a
   \]

3. Move conjunction upward, using this equivalence:
   
   \[
   a \lor (\beta \land \gamma) \quad \text{changes to} \quad (a \lor \beta) \land (a \lor \gamma)
   \]

4. Collect terms:
   
   \[
   a \lor a \quad \text{changes to} \quad a
   \]
Because CNF is entirely regular, we can omit the logical operators and write it as a set of clauses:

\[
\{(p, \neg q), \{r\}, \{s\}\}
\]
Resolution inference rule:

\[
A_1 \lor \cdots \lor A_n \lor B, \neg B \lor C_1 \lor \cdots \lor C_m \]

\[
\frac{A_1 \lor \cdots \lor A_n \lor B, \neg B \lor C_1 \lor \cdots \lor C_m}{A_1 \lor \cdots \lor A_n \lor C_1 \lor \cdots \lor C_m}
\]

Each $A_i$ or $C_j$ is a propositional variable or its negation.
To determine if a knowledge base $\Delta$ entails a proposition $\psi$: (i.e., does $\Delta \vdash \psi$):

1. Put $\Delta$ and the negation of the goal, $\neg \psi$, into CNF to get a set of clauses, $S$.
2. Check if $\emptyset$ is in $S$. If so, $\Delta \vdash \psi$.
   
   This means you found a contradiction, e.g., $\alpha$ and $\neg \alpha$, which you resolved to wind up with an empty clause.
3. Check if there are two clauses in $S$ that resolve to produce a clause not already in $S$. If not, $\Delta \not\vdash \psi$.
4. Add the new clause to $S$ and go to Step 2.
Exercise

Knowledge base:

\[ \text{Rain} \lor \text{Sun} \]

\[ \text{Sun} \Rightarrow \text{Mail} \]

\[ (\text{Rain} \lor \text{Sleet}) \Rightarrow \text{Mail} \]

Is there Mail?
Convert to CNF:

1. Replace any ⇒ or ⇔ by equivalent formulas using ∧, ∨, and ¬:
   - \( a \Rightarrow \beta \) changes to \( \neg a \lor \beta \)
   - \( a \Leftrightarrow \beta \) changes to \( (a \land \beta) \lor (\neg a \land \neg \beta) \)

2. Move negation inward, using de Morgan’s laws and eliminating double negations:
   - \( \neg (a \land \beta) \) changes to \( \neg a \lor \neg \beta \)
   - \( \neg (a \lor \beta) \) changes to \( \neg a \land \neg \beta \)
   - \( \neg \neg a \) changes to \( a \)

3. Move conjunction upward, using this equivalence:
   - \( a \lor (\beta \land \gamma) \) changes to \( (a \lor \beta) \land (a \lor \gamma) \)

4. Collect terms:
   - \( a \lor a \) changes to \( a \)
Convert to CNF:

1. Replace any $\Rightarrow$ or $\Leftrightarrow$ by equivalent formulas using $\land$, $\lor$, and $\neg$:

   - $a \Rightarrow \beta$ changes to $\neg a \lor \beta$
   - $a \Leftrightarrow \beta$ changes to $(a \land \beta) \lor (\neg a \land \neg \beta)$

2. Move negation inward, using de Morgan’s laws and eliminating double negations:

   - $\neg(a \land \beta)$ changes to $\neg a \lor \neg \beta$
   - $\neg(a \lor \beta)$ changes to $\neg a \land \neg \beta$
   - $\neg \neg a$ changes to $a$

3. Move conjunction upward, using this equivalence:

   - $a \lor (\beta \land \gamma)$ changes to $(a \lor \beta) \land (a \lor \gamma)$

4. Collect terms:

   - $a \lor a$ changes to $a$
Write as clauses:

\[
\begin{align*}
\{ \text{Rain, Sun} \} \\
\{ \neg \text{Sun, Mail} \} \\
\{ \neg \text{Sleet, Mail} \} \\
\{ \neg \text{Rain, Mail} \}
\end{align*}
\]
Add negation of goal:

\{¬\text{Mail}\}
Apply resolution until we get a contradiction:

\[
\{\text{Rain}, \text{Sun}\} \quad \{\neg \text{Sun}, \text{Mail}\} \quad \{\neg \text{Mail}\} \quad \{\neg \text{Rain}, \text{Mail}\}
\]

Contradiction, so \textit{Mail} was entailed!
Acknowledgments

The lecture incorporates material from:

Ronald Brachman, Cornell University
George Konidaris, Brown University
Hector Levesque, University of Toronto
Peter Norvig, Google
Stuart Russell, University of California, Berkeley; ai.berkeley.edu
Keith Schwarz, Stanford University