Lecture 5 - Propositions and Inference

Jesse Hoey
School of Computer Science
University of Waterloo
January 28, 2019

Problem Solving

Two methods for solving problems:
- **Procedural**
  - devise an algorithm
  - program the algorithm
  - execute the program
- **Declarative**
  - identify the knowledge needed
  - encode the knowledge in a representation (knowledge base - KB)
  - use logical consequences of KB to solve the problem

Proof Procedures

A logic consists of
- syntax: what is an acceptable sentence?
- semantics: what do the sentences and symbols mean?
- proof procedure: how do we construct valid proofs?

A proof: a sequence of sentences derivable using an inference rule

Logical Connectives

and (conjunction) \( \land \)
or (disjunction) \( \lor \)
not (negation) \( \neg \)
if ... then ... (implication) \( \rightarrow \)
... if and only if ... \( \leftrightarrow \)

Implication Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ( \rightarrow ) B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) If it rains, then I will carry an umbrella
Implication Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</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>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) (B)
If it rains, then I will carry an umbrella
If you don't study, then you will fail

If and only if Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ↔ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) (B)
if it rains then the grass is wet
the grass is not wet
therefore it's not raining

De Morgan's Laws

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ∨ B</th>
<th>¬(A ∧ ¬B)</th>
<th>¬A ∨ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) (B)
it rains or I play football
not true that (it doesn’t rain AND I don’t play football)
I’m a politician AND I lie
not true that (I’m not a politician OR I tell the truth)

Modus Ponens

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) ∧ A</th>
<th>((A → B) ∧ A) → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Ponens is a Tautology
If it’s raining then the grass is wet
it’s raining
therefore the grass is wet

Modus Tolens

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) ∧ ¬B</th>
<th>((A → B) ∧ ¬B) → ¬A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Tolens is a Tautology
If it’s raining then the grass is wet
the grass is not wet
therefore it’s not raining
### Modus Bogus

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) ∧ B</th>
<th>((A → B) ∧ B) → A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Bogus is not a **Tautology**

If it’s raining then the grass is wet
the grass is wet
therefore its raining

---

### Logical Consequence

A statement, A, is a logical consequence of a set of statements \{X\},
if A is true in every model of \{X\}.

If, for every set of truth assignments that hold for \{X\} (for every model of \{X\}), some other statement (A) is always true,
then this other statement is a logical consequence of \{X\}.

### Arguments and Models

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>D1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>C</td>
<td>H</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</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>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

P1: If I play hockey, then I’ll score a goal if the goalie is not good
P2: If I play hockey, the goalie is not good
D: Therefore, if I play hockey, I’ll score a goal

P: I play hockey
C: I’ll score a goal
H: the goalie is good

P1:  \( P \rightarrow (\neg H \rightarrow C) \)  
P2:  \( P \rightarrow \neg H \)  
D:  \( P \rightarrow C \)

---

### Argument Validity

An argument is valid if any of the following is true:

- the conclusions are a logical consequence of the premises.
- the conclusions are true in every model of the premises.
- there is no situation in which the premises are all true, but the conclusions are false.
- argument → conclusions is a tautology (always true)

(These four statements are identical)
Arguments and Models

<table>
<thead>
<tr>
<th>P</th>
<th>C</th>
<th>H</th>
<th>¬H → C</th>
<th>P1</th>
<th>P2</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Each row is an *interpretation*: an assignment of T/F to each proposition.

In all the green lines, the premises are true:

these interpretations are *models* of $P_1$ and $P_2$.

Every *model* of $P_1$ and $P_2$ is a *model* of $D$.

Therefore, $D$ is a *logical consequence* of $P_1$ and $P_2$:

$P_1, P_2 \models D$.

Logical Consequence

P1: Elvis is Dead
P2: Elvis is Not Dead
D: Therefore, Gerry is Alive

Is this argument valid?

Yes!

E: Elvis is Alive
G: Gerry is Alive

<table>
<thead>
<tr>
<th>E</th>
<th>¬E</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</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>T</td>
</tr>
</tbody>
</table>

An argument is *valid* if there is no situation in which the premises are all true, but the conclusions are false.

But here, there is no model of the premises, so the argument is valid.

Deduction and Proof

Given a knowledge base, we want to prove things that are true.

We can use

- Truth Table
- Natural Deduction
- Semantic Tableaux
- Axiomatic Logic (Modus Ponens)
  $((A \rightarrow B) \land A) \rightarrow B$
- Resolution Refutation (Reductio Ad Absurdum)
  $(\neg A) \land \ldots \land \ldots \rightarrow \bot) \rightarrow A$

Proofs

- A KB is a set of axioms
- A *proof procedure* is a way of Proving Theorems
  - KB $\vdash g$ means g can be *derived* from KB using the proof procedure
  - If KB $\vdash g$, then g is a *Theorem*
  - A proof procedure is *sound*:
    - if KB $\vdash g$ then KB $\models g$.
  - A proof procedure is *complete*:
    - if KB $\models g$ then KB $\vdash g$.
  - Two types of proof procedures:
    - *bottom up* and *top down*

Complete Knowledge

- we assume a *closed world*
  - the agent knows everything (or can prove everything)
  - if it can’t prove something: must be false
  - negation as failure
- other option is an *open world*:
  - the agent doesn’t know everything
  - can’t conclude anything from a lack of knowledge
also known as **forward chaining** - start from facts and use rules to generate all possible atoms

\[
\begin{align*}
\text{rain} & \leftarrow \text{clouds} \land \text{wind}.
\text{clouds} & \leftarrow \text{humid} \land \text{cyclone}.
\text{clouds} & \leftarrow \text{near_sea} \land \text{cyclone}.
\text{wind} & \leftarrow \text{cyclone}.
\text{near_sea}.
\text{cyclone}.
\{\text{near_sea}, \text{cyclone}\}
\{\text{near_sea}, \text{cyclone}, \text{wind}\}
\end{align*}
\]

also known as **forward chaining** - start from facts and use rules to generate all possible atoms

\[
\begin{align*}
\text{rain} & \leftarrow \text{clouds} \land \text{wind}.
\text{clouds} & \leftarrow \text{humid} \land \text{cyclone}.
\text{clouds} & \leftarrow \text{near_sea} \land \text{cyclone}.
\text{wind} & \leftarrow \text{cyclone}.
\text{near_sea}.
\text{cyclone}.
\{\text{near_sea}, \text{cyclone}\}
\{\text{near_sea}, \text{cyclone}, \text{wind}\}
\end{align*}
\]

\[
C := \{\};
\text{repeat}
\begin{align*}
\text{select } r & \in KB \text{ such that }
\cdot r \text{ is } h \leftarrow b_1 \land \ldots \land b_m \\
\cdot b_i & \in C \quad \forall \ i
\cdot h \notin C
\quad C := C \cup \{h\}
\end{align*}
\text{until no more clauses can be selected}
\]

Sound and Complete
Top-Down Proof (no variables)

start from query and work backwards

rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.

yes ← rain.

yes ← clouds ∧ wind

yes ← near_sea ∧ cyclone ∧ wind

yes ← near_sea ∧ cyclone ∧ cyclone
Top-Down Proof (no variables)

start from query and work backwards

rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.

yes ← rain.
yes ← clouds ∧ wind
yes ← near_sea ∧ cyclone ∧ wind
yes ← near_sea ∧ cyclone ∧ cyclone
yes ← near_sea ∧ cyclone
yes ← cyclone

Top-Down Interpreter (no variables)

solve(q1 ∧ ... ∧ qk):
    ac := “yes ← q1 ∧ ... ∧ q′′
    repeat
        select a conjunct ai from body of ac
        choose a clause C from KB with ai as head
        replace ai in body of ac by body of C
    until ac is an answer

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
water.
bread.

yes ← health.

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
water.
bread.

yes ← health.
yes ← sustenance ∧ shelter
Top-Down Proof (no variables)

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
water.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← bread ∧ water ∧ shelter

Top-Down Proof (no variables)

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
water.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← bread ∧ water ∧ shelter
yes ← bread ∧ water ∧ house
yes ← bread ∧ water

Top-Down Interpreter (no variables)

solve(q₁ ∧...∧qₖ):
   \(ac := \text{"yes ← q₁ ∧...∧ qₖ"}\)
   \(\text{repeat}\)
   \(\text{select} \) a conjunct \(a_i\) from body of \(ac\)
   \(\text{choose} \) a clause \(C\) from KB with \(a_i\) as head
   replace \(a_i\) in body of \(ac\) by body of \(C\)
   until \(ac\) is an answer

select: “don’t care nondeterminism”
(If one doesn’t give a solution, no point trying others!)
choose: “don’t know nondeterminism”
(if one doesn’t give a solution, others may)
Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter

Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← macdonalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
Top-Down Proof - selection failure

start from query and work backwards

health ← sustenance ∧ shelter.
sustenance ← maconalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← bread ∧ water ∧ shelter
yes ← bread ∧ water ∧ house
yes ← bread ∧ water
yes ← water

Failure with this choice, try the other
(have to try them all)

Top-Down Proof - selection failure

start again part way through

health ← sustenance ∧ shelter.
sustenance ← maconalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← maconalds ∧ cocacola ∧ shelter
yes ← maconalds ∧ cocacola ∧ house
yes ← maconalds ∧ cocacola

Top-Down Proof - selection failure

start again part way through

health ← sustenance ∧ shelter.
sustenance ← maconalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← maconalds ∧ cocacola ∧ shelter
yes ← maconalds ∧ cocacola ∧ house
yes ← maconalds ∧ cocacola

Top-Down Proof - selection failure

start again part way through

health ← sustenance ∧ shelter.
sustenance ← maconalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← maconalds ∧ cocacola ∧ shelter
yes ← maconalds ∧ cocacola ∧ house
yes ← maconalds ∧ cocacola

Top-Down Proof - selection failure

start again part way through

health ← sustenance ∧ shelter.
sustenance ← maconalds ∧ cocacola.
sustenance ← bread ∧ water.
shelter ← house.
house.
cocacola.
bread.

yes ← health.
yes ← sustenance ∧ shelter
yes ← maconalds ∧ cocacola ∧ shelter
yes ← maconalds ∧ cocacola ∧ house
yes ← maconalds
Refutation of Modus Ponens

\[ A \land (A \rightarrow B) \vdash B \]

show a contradiction

\[ A \land (A \rightarrow B) \land \neg B \vdash \bot \]

1. \[ A \land (\neg A \lor B) \land (\neg B) \]
2. \[ \{A\}, \{\neg A, B\}, \{\neg B\} \]

Transitivity of Implication

\[ ((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C) \]

try to show a contradiction

\[ (A \rightarrow B) \land (B \rightarrow C) \land \neg(A \rightarrow C) \vdash \bot \]

1. \[ (\neg A \lor B) \land (\neg B \lor C) \land (\neg \neg A \land \neg C) \]
2. \[ (\neg A \lor B) \land (\neg B \lor C) \land (\neg \neg A \land \neg C) \]
3. \[ (\neg A \lor B) \land (\neg B \lor C) \land A \land \neg C \]
4. \[ \{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\} \]
A complementary pair of propositions is $p_i, \neg p_i$

- Can show that two clauses with a complementary pair:
  $\{\{A, B\}, \{C, \neg B\}\} \equiv \{\{A, B\}, \{C, \neg B\}, \{A, C\}\}$

- That is, since $B$ and $\neg B$ cannot both be true, one of $A$ or $C$ has to be true, otherwise the whole formula is false

- Therefore, we can "resolve" $\{A, B\}, \{C, \neg B\}$ into $\{A, C\}$

- This means that $\{A, C\}$ is true whenever $\{A, B\}, \{C, \neg B\}$ is true

- So we can add $\{A, C\}$ to the statement without changing the truth value

**Resolution**

- Proof by *resolution refutation*: deny the conclusions and show a resolution to $\bot$.
- Resolve clauses - adds new clauses that are true whenever the existing clauses are true
- If you can find a contradiction, then
  - the existing clauses cannot all be true
  - If the premises are all true, the refutation of the conclusion must be false,
    - so the argument is valid
- If you cannot find a contradiction after resolving all clauses
  - the refutation of the conclusion must be true
  - so the argument is invalid

---

**Resolution - Example 3**

Refutation of Modus Ponens

$A \land (A \rightarrow B) \vdash B$

show a contradiction

$A \land (A \rightarrow B) \land \neg B \vdash \bot$

1. $A \land (\neg A \lor B) \land (\neg B)$
2. $\{\{A\}, \{\neg A, B\}, \{\neg B\}\}$
3. $\{\{A\}, \{\neg A, B\}, \{B\}, \{\neg B\}\}$
4. $\bot$

**Resolution - Example 2**

Transitivity of Implication (again)

$((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C)$

try to show a contradiction

$(A \rightarrow B) \land (B \rightarrow C) \land \neg(A \rightarrow C) \vdash \bot$

1. $\neg(A \lor B) \land (\neg B \lor C) \land (\neg A \lor C)$
2. $\neg(A \lor B) \land (\neg B \lor C) \land (\neg A \land \neg C)$
3. $(\neg A \lor B) \land (\neg B \lor C) \land (\neg A \land \neg C)$
4. $\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}$
5. $\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{\neg C\}$
6. $\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{C\}, \{\neg C\}$
7. $\bot$

---

**Resolution - Example 1**

Proof by resolution refutation: deny the conclusions and show a resolution to $\bot$.

Resolve clauses - adds new clauses that are true whenever the existing clauses are true

- If you can find a contradiction, then
  - the existing clauses cannot all be true
  - If the premises are all true, the refutation of the conclusion must be false,
    - so the argument is valid
- If you cannot find a contradiction after resolving all clauses
  - the refutation of the conclusion must be true
  - so the argument is invalid

**Combinatorial Search Problems**

- Many problems can be formulated as a CNF
  - Satisfiability
  - Logic circuits
  - Gene decoding
  - Scheduling
  - Air traffic control
  - ...
A CSP variable $Y$ with domain $\{v_1, \ldots, v_k\}$ can be converted into $k$ Boolean variables $\{Y_1, \ldots, Y_k\}$ where $Y_i$ is true when $Y$ has value $v_i$ and false otherwise.

Thus, $k$ atoms $y_1, \ldots, y_k$ are used to represent the CSP variable.

Constraints:
- $y_i$ and $y_j$ cannot both be true when $i \neq j$: $\neg y_i \lor \neg y_j$ for $i < j$.
- At least one of the $y_i$ must be true: $y_1 \lor \ldots \lor y_k$.
- There is a clause for each false assignment in each constraint, which specifies which assignments are not allowed.
- Thus, if there are two variables $Y$ and $Z$, and a constraint $Y \neq Z$, then we have clauses $\neg y_i \lor \neg z_i$ for all $i$ (Assuming $Y$ and $Z$ have the same domains).

Example Delivery robot: activities $a, b$, times $1, 2, 3, 4$.

Constraints:
- $(A \neq 2) \land (B \neq 1) \land (A < B)$

Write down the CNF for this CSP.

Where propositions stop...
- $\text{peter is smart } \land \text{peter is a student } \rightarrow \text{peter will pass}$
- $\text{ann is smart } \land \text{ann is a student } \rightarrow \text{ann will pass}$
- $\text{lou is smart } \land \text{lou is a student } \rightarrow \text{lou will pass}$

$\text{peter is smart and peter is a student}$
$\text{peter will pass}$

Where propositions stop...
- $\text{ann is smart } \land \text{ann is a student } \rightarrow \text{ann will pass}$
- $\text{lou is smart } \land \text{lou is a student } \rightarrow \text{lou will pass}$

$\text{lou is smart and lou is a student}$
$\text{lou will pass}$
Where propositions stop...

\begin{align*}
peter & \text{ is smart } \land \text{ peter is a student} \rightarrow \text{ peter will pass} \\
ann & \text{ is smart } \land \text{ ann is a student} \rightarrow \text{ ann will pass} \\
lou & \text{ is smart } \land \text{ lou is a student} \rightarrow \text{ lou will pass} \\
\text{All smart students pass} \\
\text{ann is smart and ann is a student} \\
\text{ann will pass} \\
\forall X(\text{smart}(X) \land \text{ student}(X) \rightarrow \text{will pass}(X))
\end{align*}

Where propositions stop...

\begin{align*}
peter & \text{ is smart } \land \text{ peter is a student} \rightarrow \text{ peter will pass} \\
ann & \text{ is smart } \land \text{ ann is a student} \rightarrow \text{ ann will pass} \\
lou & \text{ is smart } \land \text{ lou is a student} \rightarrow \text{ lou will pass} \\
\text{All smart students pass} \\
\text{ann is smart and ann is a student} \\
\text{ann will pass} \\
\text{smart}(\text{ann}) \land \text{ student}(\text{ann}) \rightarrow \text{ will pass}(\text{ann}) \\
\forall X(\text{smart}(X) \land \text{ student}(X) \rightarrow \text{ will pass}(X))
\end{align*}

Beyond propositions: Individuals and Relations

\text{MIU Puzzle}

\begin{itemize}
\item Symbols: M, I, U
\item Axiom: MI
\item Rules:
\begin{itemize}
\item if \( xI \) is a theorem, so is \( xIU \)
\item \( Mx \) is a theorem, so is \( Mxx \)
\item in any theorem, \( III \) can be replaced by \( U \)
\item \( UU \) can be dropped from any string
\end{itemize}
\item Starting from MI, can you generate MU? (use top-down or bottom-up)
\end{itemize}
Planning under certainty (Poole & Mackworth 2nd ed. Chapter 6.1-6.4)

Supervised Learning (Poole & Mackworth 2nd ed. Chapter 7.1-7.6)