Lecture 5 - Propositions and Inference

Jesse Hoey
School of Computer Science
University of Waterloo

May 16, 2022

Readings: Poole & Mackworth 2nd ed. chapter 5.1-5.3, and 13.1-13.2

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

Problem Solving

Two methods for solving problems:

- Procedural
  - "how to" knowledge
  - programs
  - meaning of symbols is meaning of computation
  - languages: C,C++,Java ...
- Declarative
  - descriptive knowledge
  - databases
  - meaning of symbols is meaning in world
  - languages: propositional logic, Prolog, relational databases, ...

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*)	$\bigwedge$
or (*disjunction*)	$\bigvee$
not (*negation*)	$\neg$
if ... then ... (implication)	$\rightarrow$
... if and only if ...	$\leftrightarrow$

Note: often logical statements with implication are written backwards: $A \rightarrow B$ is the same as $B \leftarrow A$.

Implication Truth Table

A	B	$A \to B$
F	F	T
F	T	T
T	F	F
T	T	T

(A) (B)
If it rains, then I will carry an umbrella

Implication Truth Table

A	B	$A \rightarrow B$
F	F	T
F	T	T
T	F	F
T	T	T

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

Implication Truth Table

A	B	$A \rightarrow B$	$A \wedge \neg B$	$\neg(A \wedge \neg B)$	$\neg A \vee B$
F	F	T	F	T	T
F	T	T	F	T	T
T	F	F	T	F	F
T	T	T	F	T	T

(A) (B)

no rain or I will carry an umbrella

study or you will fail

If and only if Truth Table

A	B	$A \leftrightarrow B$
F	F	T
F	T	F
T	F	F
T	T	T

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

De Morgan's Laws

$$A \vee B \equiv \neg(\neg A \wedge \neg B)$$

it rains OR I play football

not true that (it doesn't rain AND I don't play football)

$$A \wedge B \equiv \neg(\neg A \vee \neg B)$$

I'm a politician AND I lie

not true that (I'm not a politician OR I tell the truth)

Modus Ponens

A	B	$A \to B$	$(A \to B) \land A$	$((A \to B) \land A) \to B$
F	F	T	F	T
F	T	T	F	T
T	F	F	F	T
T	T	T	T	T

Modus Ponens is a Tautology

If it's raining then the grass is wet

it's raining

therefore the grass is wet

Modus Tolens

A	B	$A \rightarrow B$	$(A \rightarrow B) \wedge \neg B$	$((A \rightarrow B) \wedge \neg B) \rightarrow \neg A$
F	F	T	T	T
F	T	T	F	T
T	F	F	F	T
T	T	T	F	T

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

A	B	$A \to B$	$(A \to B) \land B$	$((A \to B) \land B) \to A$
F	F	T	F	T
F	T	T	T	F
T	F	F	F	T
T	T	T	T	T

Modus Bogus is not a Tautology
If it's raining then the grass is wet
the grass is wet
therefore its raining

Logical Consequence

- $\{X\}$ is a set of statements
- A set of truth assignments to $\{X\}$ is an interpretation
- A model of $\{X\}$ is an interpretation that makes $\{X\}$ true.
- We say that the world in which these truth assignments hold is a model (a verifiable example) of $\{X\}$.
- $\{X\}$ is inconsistent if it has no model

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 $\{X\}$

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 $\rightarrow$ conclusions is a tautology (always true)

(these four statements are identical)

Arguments and Models

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

$$P1 : P \to (\neg H \to C) \quad P2 : P \to \neg H$$
$$D : P \to C$$

P	C	H	$\neg H \to C$	$P1$	$P2$	D
F	F	F	F	T	T	T
F	F	T	T	T	T	T
F	T	F	T	T	T	T
F	T	T	T	T	T	T
T	F	F	F	F	T	F
T	F	T	T	T	F	F
T	T	F	T	T	T	T
T	T	T	T	T	F	T

Arguments and Models

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

$$P1 : P \to (\neg H \to C) \quad P2 : P \to \neg H$$
$$D : P \to C$$

P	C	H	$\neg H \to C$	$P1$	$P2$	D
F	F	F	F	T	T	T
F	F	T	T	T	T	T
F	T	F	T	T	T	T
F	T	T	T	T	T	T
T	F	F	F	F	T	F
T	F	T	T	T	F	F
T	T	F	T	T	T	T
T	T	T	T	T	F	T

Arguments and Models

P	C	H	$\neg H \to C$	$P1$	$P2$	D
F	F	F	F	T	T	T
F	F	T	T	T	T	T
F	T	F	T	T	T	T
F	T	T	T	T	T	T
T	F	F	F	F	T	F
T	F	T	T	T	F	F
T	T	F	T	T	T	T
T	T	T	T	T	F	T

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 $P1$ and $P2$.

Every model of $P1$ and $P2$ is a model of D.

Therefore, D is a logical consequence of $P1$ and $P2$:

$$P1, P2 \models D.$$

Logical Consequence

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

Is this argument valid?

Logical Consequence

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

Is this argument valid?
Yes!
E: Elvis is Alive
J: Jerry is Alive

E	¬ E	J
F	T	F
F	T	T
T	F	F
T	F	T

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)
  $((A \rightarrow B) \wedge A) \rightarrow B$
- Resolution Refutation (Reductio Ad Absurdum)
  $(\neg A) \wedge \ldots \wedge \ldots \rightarrow \bot) \rightarrow A$

Proofs

- A Knowledge Base (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

Bottom-up proof

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

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

Bottom-up proof

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

```
rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.
{near_sea,cyclone }
```

Bottom-up proof

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

```
rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.
{near_sea,cyclone }
{near_sea ,cyclone ,wind }
```

Bottom-up proof

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

```
rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.
{near_sea,cyclone }
{near_sea ,cyclone ,wind }
{near_sea ,cyclone ,wind ,clouds }
```

Bottom-up proof

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

```
rain ← clouds ∧ wind.
clouds ← humid ∧ cyclone.
clouds ← near_sea ∧ cyclone.
wind ← cyclone.
near_sea.
cyclone.
{near_sea,cyclone }
{near_sea ,cyclone ,wind }
{near_sea ,cyclone ,wind ,clouds }
{near_sea ,cyclone ,wind ,clouds ,rain }
```

Bottom-up proof

$C := \{\};$
repeat
    select $r \in KB$ such that
        $\cdot\ r$ is $h \leftarrow b_1 \wedge \ldots \wedge b_m$
        $\cdot\ b_i \in C \quad \forall \quad i$
        $\cdot\ h \notin C$
    $C := C \cup \{h\}$
until no more clauses can be selected

Sound and Complete

Top-Down Proof

start from query and work backwards

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

Start with query: if `rain` is proved, "yes" is the logical result (the answer to the question)

Top-Down Proof

start from query and work backwards

$\texttt{rain} \leftarrow \texttt{clouds} \wedge \texttt{wind}.$
$\texttt{clouds} \leftarrow \texttt{humid} \wedge \texttt{cyclone}.$
$\texttt{clouds} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}.$
$\texttt{wind} \leftarrow \texttt{cyclone}.$
$\texttt{near\_sea}.$
$\texttt{cyclone}.$

Start with query: if rain is proved, "yes" is the logical result (the answer to the question)
$\texttt{yes} \leftarrow \texttt{rain}.$

Top-Down Proof

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
```

Top-Down Proof

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
```

Top-Down Proof

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

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
```

Top-Down Proof

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 Proof

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
yes ←
```

Top-Down Interpreter

solve($q_1 \wedge \ldots \wedge q_k$):
  $ac :=$ "$yes \leftarrow q_1 \wedge \ldots \wedge q_k$"
  repeat
    select a conjunct q_i from body of ac
    choose a clause C from KB with q_i as head
    replace q_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!
  any one will do, but be careful: some selections will lead more quickly to solutions!

choose: "don't know nondeterminism"
  if one doesn't give a solution, others may
  have to do them all: can determine complexity of the problem

Towards Automated Methods

- A proof procedure gives us a method for deriving theorems
- Therefore, given a knowledge base of assumptions, we can 'prove' things and know they are tautologies (they are logical consequences of our knowledge base)

but ....

The method is difficult and requires some know-how - how could we make it work more automatically?

Conjunctive Normal Form

A well-formed formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions of atoms.

$(p_1 \vee p_2) \wedge (p_3 \vee p_4 \vee p_5) \wedge (p_6 \vee p_7 \vee ...)... \wedge (p_{n-1} \vee p_n)$

Convert a propositional formula to CNF:

1. Eliminate $\leftrightarrow$ using $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$
2. Eliminate $\rightarrow$ using $A \rightarrow B \equiv \neg A \vee B$
3. Use deMorgan's laws to push $\neg$ into atoms
4. Use $\neg\neg A \equiv A$ to eliminate double negatives
5. use distributive law to complete
   $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

write
$(p_1 \vee p_2) \wedge (p_3 \vee p_4 \vee p_5) \wedge (p_6 \vee p_7 \vee ...)... \wedge (p_{n-1} \vee p_n)$
as
$\{\{p_1, p_2\}, \{p_3, p_4, p_5\}, \{p_6, p_7, ...\}..., \{p_{n-1}, p_n\}\}$

Conjunctive Normal Form - Example 1

Refutation of Modus Ponens

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

show a contradiction $\bot$: means "false"

If our refutation leads to a contradiction, it must be "false", so the conclusion must be true

$A \wedge (A \rightarrow B) \wedge \neg B \models \bot$

1. $A \wedge (\neg A \vee B) \wedge (\neg B)$
2. $\{\{A\}, \{\neg A, B\}, \{\neg B\}\}$

can already tell this is false since A must be true, so B must be true, but B must be false

We will demonstrate using resolution on slide 28

Conjunctive Normal Form - Example 2

Transitivity of Implication

$((A \to B) \wedge (B \to C)) \to (A \to C)$

try to show a contradiction

$(A \to B) \wedge (B \to C) \wedge \neg(A \to C) \models \bot$

1. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge \neg(\neg A \vee C)$
2. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg\neg A \wedge \neg C)$
3. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge A \wedge \neg C$
4. $\{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}$

Resolution

- 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
- $\{\{A\}, \{\neg A\}\}$ resolves to $\bot$

Resolution

- Proof by resolution refutation: deny the conclusions and show a resolution to $\perp$.
- 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 1

Refutation of Modus Ponens

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

show a contradiction

$A \wedge (A \rightarrow B) \wedge \neg B \models \bot$

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

Resolution - Example 2

Transitivity of Implication (again)

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

try to show a contradiction

$(A \rightarrow B) \wedge (B \rightarrow C) \wedge \neg(A \rightarrow C) \models \bot$

1. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge \neg(\neg A \vee C)$
2. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg\neg A \wedge \neg C)$
3. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge A \wedge \neg C$
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 3

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: if I play hockey , I'll score a goal

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

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

$D : P \rightarrow C$ test (refutation of D): $P1 \wedge P2 \wedge \neg D$

$$(P \rightarrow (\neg H \rightarrow C)) \wedge (P \rightarrow \neg H) \wedge \neg(P \rightarrow C)$$

$$(\neg P \vee (\neg H \rightarrow C)) \wedge (\neg P \vee \neg H) \wedge \neg(\neg P \vee C)$$

$$(\neg P \vee (\neg\neg H \vee C)) \wedge (\neg P \vee \neg H) \wedge \neg(\neg P \vee C)$$

$$(\neg P \vee \neg\neg H \vee C) \wedge (\neg P \vee \neg H) \wedge (\neg\neg P \wedge \neg C)$$

$$(\neg P \vee H \vee C) \wedge (\neg P \vee \neg H) \wedge (P) \wedge (\neg C)$$

$$\{\{\neg P, H, C\}, \{\neg P, \neg H\}, \{P\}, \{\neg C\}\}$$

$$\{\{\neg P, H, C\}, \{\neg P, \neg H\}, \{\neg P, C\}, \{P\}, \{\neg C\}\}$$

$$\{\{\neg P, H, C\}, \{\neg P, \neg H\}, \{\neg P, C\}, \{P\}, \{C\}, \{\neg C\}\}$$

CNF for weather example

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

prove `rain` by converting to CNF and resolving

Combinatorial Search Problems

Many problems can be formulated as a CNF

- Satisfiability
- Logic circuits
- Gene decoding
- Scheduling
- Air traffic control
- ...

Constraint Satisfaction as CNF

- 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:
  - exactly one of $y_1, \ldots, y_k$ must be true:
    - y_i and y_j cannot both be true when $i \neq j$: $\neg y_i \vee \neg y_j$ for $i < j$
    - at least one of the y_i must be true: $y_1 \vee \ldots \vee y_k$
  - There is a clause for each false assignment in each constraint that 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 $\neg y_i \vee \neg z_i$ for all i (Assuming Y and Z have the same domains).

Constraint Satisfaction as CNF

Example Delivery robot: activities a,b , times 1,2,3,4 .
constraints :
$(A \neq 2) \wedge (B \neq 1) \wedge (A < B)$
We have two 8 variables in the CNF:

$$a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4$$

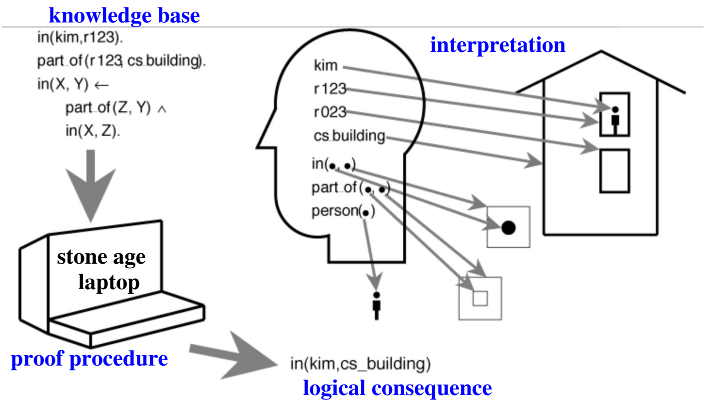
where a_i means $A = i$ is true and b_i means $B = i$ is true.
Constraints saying that A (and B) must be exactly one value:

$$\neg a_i \vee \neg a_j \quad \text{for} \quad i < j \qquad\qquad a_1 \vee a_2 \vee a_3 \vee a_4$$

$$\neg b_i \vee \neg b_j \quad \text{for} \quad i < j \qquad\qquad b_1 \vee b_2 \vee b_3 \vee b_4$$

Domain constraints $\neg a_2$ and $\neg b_1$
The binary constraint $A < B$ has one $\neg(a_i \wedge b_j)$ for all $j \leq i$

Beyond propositions: Individuals and Relations



- KB can contain relations : `part_of(C,A)` is true if C is a "part of" A (in the world)
- KB can contain quantification : `part_of(C,A)` holds $\forall C, A$
- proof procedure is the same, with a few extra bits to handle relations & quantification

First order example

```
symptom(runny_nose,flu).
symptom(fever,flu).
symptom(fever,hepatitis).
symptom(chills,flu).
symptom(chills,hypothermia).
symptom(aches,flu).
symptom(rash,hepatitis).

has_symptom(john,fever).
has_symptom(john,runny_nose).
has_symptom(mary,chills).
has_symptom(mary,rash).

has_condition(Person,Condition):-
      symptom(Symptom,Condition),
      has_symptom(Person,Symptom).
```

MIU Puzzle

- Symbols: M,I,U
- Axiom: MI
- Rules:
  - if x I is a theorem, so is x IU
  - M x is a theorem, so is M xx
  - in any theorem, III can be replaced by U
  - UU can be dropped from any string
- Starting from MI, can you generate MU? (use top-down or bottom-up)

Next:

- Planning under certainty (Poole & Mackworth 2nd ed. Chapter 6.1-6.4)
- Supervised Learning (Poole & Mackworth 2nd ed. Chapter 7.1-7.6)