CS486/686: Introduction to Artificial Intelligence
Lecture 5 - Inference and Planning

Jesse Hoey & Victor Zhong

School of Computer Science, University of Waterloo

January 23, 2025

Readings: Poole & Mackworth Chap. 5.1-5.3 and 6.1-6.3

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)	$\wedge$
or (disjunction)	$\vee$
not (negation)	$\neg$
if ... then ... (**implication**)	$\rightarrow$
... if and only if ...	$\iff$

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

(A) (B)

If it rains, then I will carry an umbrella

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

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 $\{X\}$

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

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

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

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

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

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)

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

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 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	$\neg$ 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**

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**
- Another 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

$\texttt{rain} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{clouds} \leftarrow \texttt{humid} \wedge \texttt{cyclone}$

$\texttt{clouds} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{wind} \leftarrow \texttt{cyclone}$

$\texttt{near\_sea}$

$\texttt{cyclone}$

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

$\texttt{rain} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{clouds} \leftarrow \texttt{humid} \wedge \texttt{cyclone}$

$\texttt{clouds} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{wind} \leftarrow \texttt{cyclone}$

$\texttt{near\_sea}$

$\texttt{cyclone}$

$\{\texttt{near\_sea}, \texttt{cyclone}\}$

$\{\texttt{near\_sea}, \texttt{cyclone}, \texttt{wind}\}$

Bottom-Up Proof

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

$\texttt{rain} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{clouds} \leftarrow \texttt{humid} \wedge \texttt{cyclone}$

$\texttt{clouds} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{wind} \leftarrow \texttt{cyclone}$

near_sea

cyclone

$\{\texttt{near\_sea}, \texttt{cyclone}\}$

$\{\texttt{near\_sea}, \texttt{cyclone}, \texttt{wind}\}$

$\{\texttt{near\_sea}, \texttt{cyclone}, \texttt{wind}, \texttt{clouds}\}$

Bottom-Up Proof

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

$\texttt{rain} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{clouds} \leftarrow \texttt{humid} \wedge \texttt{cyclone}$

$\texttt{clouds} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{wind} \leftarrow \texttt{cyclone}$

$\texttt{near\_sea}$

$\texttt{cyclone}$

$\{\texttt{near\_sea},\ \texttt{cyclone}\}$

$\{\texttt{near\_sea},\ \texttt{cyclone},\ \texttt{wind}\}$

$\{\texttt{near\_sea},\ \texttt{cyclone},\ \texttt{wind},\ \texttt{clouds}\}$

$\{\texttt{near\_sea},\ \texttt{cyclone},\ \texttt{wind},\ \texttt{clouds},\ \texttt{rain}\}$

Bottom-up proof

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

Sound and Complete

Top-Down Proof

Start from query and work backwards

`rain` $\leftarrow$ `clouds` $\wedge$ `wind`
`clouds` $\leftarrow$ `humid` $\wedge$ `cyclone`
`clouds` $\leftarrow$ `near_sea` $\wedge$ `cyclone`
`wind` $\leftarrow$ `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

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \land \texttt{wind}$

Top-Down Proof

Start from query and work backwards

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{wind}$

Top-Down Proof

Start from query and work backwards

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{cyclone}$

Top-Down Proof

Start from query and work backwards

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{cyclone}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

Top-Down Proof

Start from query and work backwards

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{cyclone}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{yes} \leftarrow \texttt{cyclone}$

Top-Down Proof

Start from query and work backwards

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

$\texttt{yes} \leftarrow \texttt{rain}$

$\texttt{yes} \leftarrow \texttt{clouds} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{wind}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone} \wedge \texttt{cyclone}$

$\texttt{yes} \leftarrow \texttt{near\_sea} \wedge \texttt{cyclone}$

$\texttt{yes} \leftarrow \texttt{cyclone}$

$\texttt{yes} \leftarrow$

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

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

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**?
  You may use either top-down **or** bottom-up proof

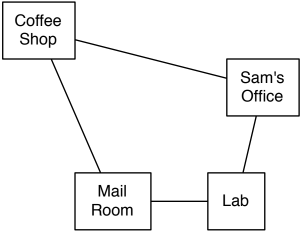
Planning

- Planning is **deciding what to do** based on an agent's ability, its goals, and the state of the world
- Planning is finding a **sequence of actions to solve a goal**
- Initial assumptions:
  - A **single** agent
  - The world is **deterministic**
  - There are **no exogenous events** outside of the control of the agent that change the state of the world
  - The agent knows what state it is in (full **observability**)
  - **Time progresses discretely** from one state to the next
  - **Goals are predicates** of states that need to be achieved or maintained (no complex goals)

Actions

- A deterministic **action** is a **partial function from states to states**
- **partial** function: some actions not possible in some states
- The **preconditions** of an action specify when the action can be carried out
- The **effect** of an action specifies the resulting state

Delivery Robot Example



Features (Variables):

RLoc – Rob's location

(4-valued: {cs,off,mr,lab})

RHC – Rob has coffee (binary)

SWC – Sam wants coffee (binary)

MW – Mail is waiting (binary)

RHM – Rob has mail (binary)

Actions:

mc – move clockwise

mcc – move counterclockwise

puc – pickup coffee

dc – deliver coffee

pum – pickup mail

dm – deliver mail

Explicit State-Space Representation

State	Action	Resulting State
$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$	mc	$\langle mr, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$	mcc	$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	dm	$\langle off, \neg rhc, swc, \neg mw, \neg rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	mcc	$\langle cs, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	mc	$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$
...	...	...

Feature-Based Representation of Actions

For each action:

- **precondition** is a proposition that specifies when the action can be carried out.

For each feature:

- **causal rules** that specify when the feature gets a new value and
- **frame rules** that specify when the feature keeps its value.

Notation:

- Features are capitalized (e.g. $Rloc$, RHC)
- Values of the features are not (e.g. $Rloc = cs$, rhc, $\neg rhc$)
- If X is a feature, then X' is the feature after an action is carried out

Example feature-based representation

Precondition of pick-up coffee (*puc*):

$$RLoc = cs \wedge \neg rhc$$

Rules for location is *cs* (specifies RLoc'):

$$RLoc' = cs \leftarrow RLoc = off \wedge Act = mcc$$
$$RLoc' = cs \leftarrow RLoc = mr \wedge Act = mc$$
$$RLoc' = cs \leftarrow RLoc = cs \wedge Act \neq mcc \wedge Act \neq mc$$

Rules for "robot has coffee" (specifies rhc'):

$$\text{(frame rule) } RHC' = true \leftarrow RCH = true \wedge Act \neq dc$$
$$(\text{or } rhc' \leftarrow rhc \wedge Act \neq dc)$$
$$\text{(causal rule) } RHC' = true \leftarrow Act = puc (\text{or } rhc' \leftarrow Act = puc)$$

Planning

Given:

- A description of the effects and preconditions of the actions
- A description of the initial state
- A goal to achieve

Find a sequence of actions that is possible and will result in a state satisfying the goal

Forward Planning

Idea: search in the state-space graph

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state s represent all of the actions that are legal in state s
- A plan is a path from the state representing the initial state to a state that satisfies the goal.
- Can use any of the search techniques from Lecture 3
- **heuristics** important
- A tutorial by Malte Helmert on Heuristics for Deterministic Planning: `https://ai.dmi.unibas.ch/misc/tutorial_aaai2015/`

Example State-Space Graph

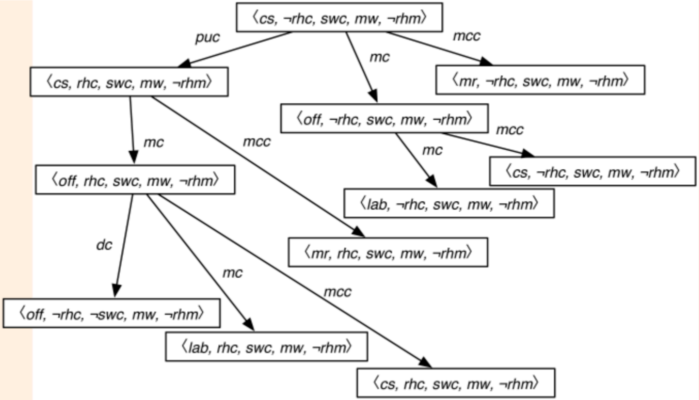


Figure 6.2: Part of the search space for a state-space planner

Regression Planning

Idea: search **backwards** from the goal description: nodes correspond to subgoals, and arcs to actions

- Nodes are propositions: a formula made up of assignments of values to features
- Arcs correspond to actions that can achieve one of the goals
- Neighbors of a node N associated with arc A specify what must be true immediately before A so that N is true immediately after
- The start node is the goal to be achieved
- goal(N) is true if N is a proposition that is true of the initial state

Next

- Supervised learning (Poole & Mackworth chapter 7.1-7.3.1, 7.4-7.4.1)