Decision Procedures for Artificial Intelligence

INF656L

Alexandre Chapoutot and Sergio Mover

ENSTA Paris

2022-2023

Lecture 3: Knowledge-based agents

1. Knowledge-based agents
   - Graph k-coloring
   - Sudoku
   - Planning by propositional inference

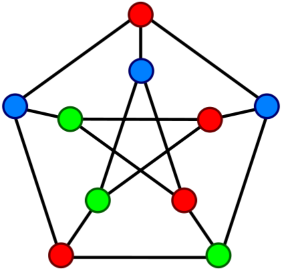
Lecture 3: Knowledge-based agents

Problem statement – graph k-coloring

Provide a propositional language and a set of axioms that formalize the graph k-coloring problem of a non-oriented graph $G = (V, E)$

Problem statement

- **Graph k-coloring problem:** A k-coloring of a non-oriented graph is a labelling of its vertices with at most k colors such that no two vertices sharing the same edge have the same color.



Logical model – Graph coloring

Language

- For all $v \in V$ and all $i \in \{1, \ldots, k\}$, p_{vi} stands for node v has color i.

Axioms

- each node has at least one color

$$\bigwedge_{v \in V} \left(\bigvee_{1 \leqslant i \leqslant k} p_{vi} \right)$$

- every node has at most one color

$$\bigwedge_{v \in V} \left(\bigwedge_{1 \leqslant i < j \leqslant k} \neg(p_{vi} \land p_{vj}) \right)$$

- adjacent nodes (v, w) do not have the same color i

$$\bigwedge_{(v,w) \in E} \left(\bigwedge_{1 \leqslant i \leqslant k} \neg(p_{vi} \land p_{wi}) \right)$$

Lecture 3: Knowledge-based agents

Problem statement – Sudoku

Sudoku is a placement puzzle. The aim of the puzzle is to enter a numeral from 1 through 9 in each cell of a grid, most frequently a 9×9 grid made up of 3×3 sub-grids (called "regions"), starting with various numerals given in some cells (the "givens"). Each row, column and region must contain only one instance of each numeral.

5		9				4		
7		8	3		4	9		
6		1				7	3	
4	6	2	5					
3	8	5	7	2		6	4	9
1		7	4		8	2		
2			1					4
		3		4			8	7
	7			5	3			6

Goal

Formalize in propositional logic of the Sudoku problem, so that any truth assignment to the propositional variables that satisfy the axioms is a solution for the puzzle.

Logical model – Sudoku

Language

- $\text{in}(n, r, c)$ stands for the number n is at row r and column c, $1 \leqslant n, r, c \leqslant 9$.

Axioms

- A row contains all numbers from 1 to 9

$$\bigwedge_{r=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{c=1}^{9} \text{in}(n, r, c)$$

- A column contains all numbers from 1 to 9

$$\bigwedge_{c=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{r=1}^{9} \text{in}(n, r, c)$$

- A region contains all numbers from 1 to 9

$$\forall 0 \leqslant k, h, \leqslant 2, \quad \bigwedge_{n=1}^{9} \bigvee_{r=1}^{3} \bigvee_{c=1}^{3} \text{in}(n, 3k + r, 3h + c)$$

- A cell cannot contain two numbers

$$\forall 1 \leqslant n, n', c, r \leqslant 9, \quad \text{with} \quad n \neq n', \quad \text{in}(n, r, c) \implies \neg\text{in}(n', r, c)$$

Lecture 3: Knowledge-based agents

SATPlan

Planning is the process of computing several steps of a problem-solving procedure before executing any of them

SATPlan

- Reduces planning problem to classical propositional SAT problem
- Can only find plans of fixed maximal length

Informally, making plans by logical inference is

1. Construct a propositional sentence that includes
   - description of the initial state
   - description of the planning domain up to some maximum time t
   - the assertion that the goal is achieved at time t
2. Call SAT solver to return a model for the sentence from 1.
   - If a model exists, extract variables representing actions at time 0 to t and are assigned true, and present them in order of times as a plan
   - If a model does not exist increase time t and go to 1

SATPlan algorithm

function SATPLAN(*init*, *actions*, *goal*, T_{max}) **returns** solution or failure
  inputs: *init*, *actions*, *goal*, constitute a description of the problem
    T_{max}, an upper limit for plan length

  for $t = 0$ **to** T_{max} **do**
    cnf ← TRANSLATE-TO-SAT(*init*, *actions*, *goal*, *t*)
    model ← SAT-SOLVER(*cnf*)
    if *model* is not *null* **then**
      return EXTRACT-SOLUTION(*model*)
  return *failure*

SATPlan vocabulary

- A *state* is a finite set of propositional variables named *fluents*.
- An *action* a is a tuple (pr, ad, de) where pr, ad and de are sets of fluents, they are denoted by Prec(a), Add(a) and Del(a) respectively. They represent the preconditions, addition effects and withdrawal effects of action a.
- A planning problem is a tuple (F, A, I, G)
  - F is a finite set of fluents
  - A is a finite set of actions based on F
  - $I \subset F$ is the initial state
  - $G \subset F$ is the goal state

SATPlan, finite horizon

- $P = (F, A, I, G)$
- For a finite horizon k, we consider that there are k successive states S_k. $S_0 = I$ and we want $S_k \supset G$. Transitions between S_i and S_{i+1} are ruled by the applications of actions at step i.
- Propositional variables
  - For each fluent $f \in F$ at step $i \in \{0, \dots, k\}$ one has the variable f_i is true iff $f \in S_i$
  - For each action $a \in A$ at step $i \in \{0, \dots, k\}$, one has the variable a_i is true iff action a is executed at step i (this implies that $Prec(a) \subseteq S_{i-1}$ and effect of a are applied on S_{i-1} to get S_i)

SATPlan, general formulation

- Initial state

$$\left(\bigwedge_{f \in I} f_0\right) \wedge \left(\bigwedge_{f \in F \setminus I} \neg f_0\right)$$

- Goal to reach

$$\bigwedge_{f \in G} f_k$$

- Preconditions and effect of actions

$$\bigwedge_{i \in \{1,\ldots,k\}} \bigwedge_{a \in A} \left[a_i \implies \left(\bigwedge_{f \in Prec(a)} f_{i-1} \wedge \bigwedge_{f \in Add(a)} f_i \wedge \bigwedge_{f \in Del(a)} \neg f_i \right) \right]$$

SATPlan, general formulation cont'd

- Frame axioms to explain removing: a fluent become false in a next state only as a consequence of the application of an action

$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{f\in(I\cup F_{add})\cap F_{del}} \left[(f_{i-1} \wedge \neg f_i) \implies \bigvee_{a\in\{b\in A | f\in Del(b)\}} a_i \right]$$

- Frame axioms to explain adding: a fluent become true in a next state only as a consequence of the application of an action

$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{f\in((F\setminus I)\cup F_{del})\cap F_{add}} \left[(\neg f_{i-1} \wedge f_i) \implies \bigvee_{a\in\{b\in A | f\in Add(b)\}} a_i \right]$$

- Interference: prevents two incompatible actions from being carried out at the same stage

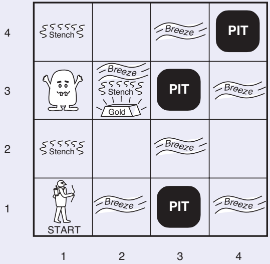
$$\bigwedge_{i\in\{1,\dots,k\}} \bigwedge_{(a,a')\in\{(b,c)\in A^2 | (b||_e c)\wedge\neg(b||_i c)\}} [\neg a_i \vee \neg a'_i]$$

where

  - $a||_e b$ iff $(Add(a) \cap Del(b)) \cup (Add(b) \cap Del(a)) = \emptyset$
  - $a||_i b$ iff $(Prec(a) \cap Del(b)) \cup (Prec(b) \cap Del(a)) = \emptyset$

Wumpus World[1] as example

Environment



- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
  Releasing drops the gold in same square

Agent's actions

- Left turn, Right turn, Forward, Grab, Release, Shoot

Agent's sensors

- Breeze, Glitter, Smell, Bump, Scream

[1]Figure from Stuart Russell and Peter Norvig, Artificial Intelligence a Modern Approach

Wumpus world: some rules

- The world if a 4x4 grids
- Agent always start at Cell $(1,1)$ oriented to East
- Agent has only one arrow
- Climb action can only be performed at Cell $(1,1)$
- Games end when the agent dies or the agent climbs out the cave

Sensors

- In a square containing the wumpus and in the directly adjacent squares (not in diagonal), agent can perceive a Stench
- In squares the directly adjacent squares to a pit, agent can perceive a Breeze
- In the square where the gold is, agent can perceive a Glitter
- When agent walks into a wall, he perceives a Bump
- When the wumpus is killed, it emits a scream that can be perceived anywhere in the cave

State of agent algorithms:

- A vector of 5 values [St, Br, Gl, Bu, Sc]

Two first steps of the game[2]

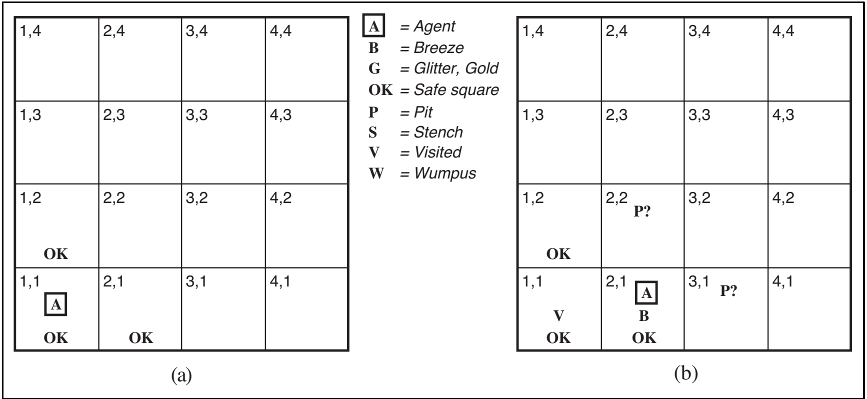


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept $[None, None, None, None, None]$. (b) After one move, with percept $[None, Breeze, None, None, None]$.

[2] Figure from Stuart Russell and Peter Norvig, Artificial Intelligence a Modern Approach

Two other steps of the game[3]

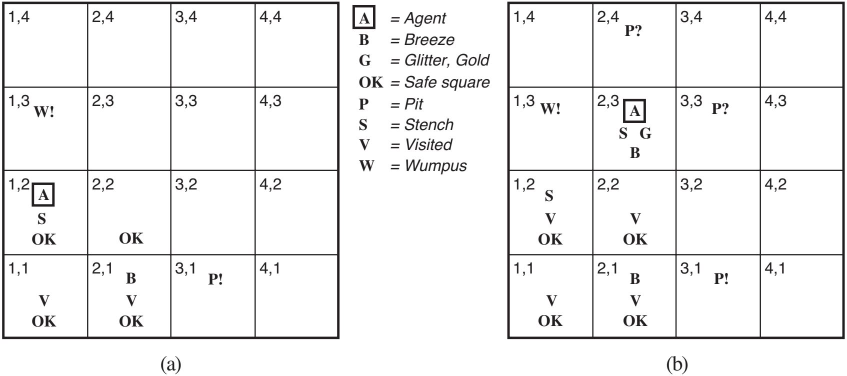


Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

[3] Figure from Stuart Russell and Peter Norvig, Artificial Intelligence a Modern Approach

Logical formulation (static)

Some propositions:

- $P_{x,y}$ is true if there is a pit at Cell (x, y)
- $W_{x,y}$ is true if there is a wumpus at Cell (x, y) (dead or alive)
- $Go_{x,y}$ is true if there is heap of gold at Cell (x, y)
- $B_{x,y}$ is true if agent perceives a Breeze at Cell (x, y)
- $S_{x,y}$ is true if agent perceives a Stench at Cell (x, y)
- $G_{x,y}$ is true if agent perceives a Glitter at Cell (x, y)

Some initial facts:

- Cell $(1, 1)$ is safe, *i.e.*, there is no pit nor wumpus

$$F_1 : \neg P_{1,1} \wedge \neg W_{1,1}$$

- If a Cell (x, y) is Breezy then there is Pit nearby

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}), B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}), \ldots \quad \text{for all cells}$$

- If a Cell (x, y) is Stenchy then there is Wumpus nearby

$$S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1}), S_{2,1} \Leftrightarrow (W_{1,1} \vee W_{2,2} \vee W_{3,1}), \ldots \quad \text{for all cells}$$

Logical formulation (static) - cont'd

Some initial facts:

- There is one and only one Wumpus in the cave

$$\text{at least one wumpus}$$
$$W_{1,1} \vee W_{2,1} \vee W_{3,1} \vee \cdots \vee W_{4,4}$$
$$\text{at most one wumpus}$$
$$(\neg W_{1,1} \vee \neg W_{1,2}) \wedge (\neg W_{1,1} \vee \neg W_{1,3}) \wedge \cdots \wedge (\neg W_{4,3} \vee \neg W_{4,4})$$

- There is one and only one heap of gold in the cave

$$\text{at least one heap of gold}$$
$$Go_{1,1} \vee Go_{2,1} \vee Go_{3,1} \vee \cdots \vee Go_{4,4}$$
$$\text{at most one heap of gold}$$
$$(\neg Go_{1,1} \vee \neg Go_{1,2}) \wedge (\neg Go_{1,1} \vee \neg Go_{1,3}) \wedge \cdots \wedge (\neg Go_{4,3} \vee \neg Go_{4,4})$$

Logical formulation (dynamic)

Some propositions are true at a given time, *e.g.*, to model **current situation**

- $L^t_{x,y}$ agent is at Cell (x, y) at time t.
- FacingEast^t, FacingWest^t, FacingNorth^t, FacingSouth^t
- HasArrow^t, agent has still an available arrow
- WumpusIsAlive^t

or to model **current perception**

- Stench^t, agent perceives a Stench at time t
- Breeze^t, agent perceives a Breeze at time t

Vocabulary

Two kinds of variables: temporal (a.k.a. *fluent*) or atemporal

Connection, for all Cells and any time:

$$L^t_{x,y} \implies (\text{Stench}^t \Leftrightarrow S_{x,y}) \quad L^t_{x,y} \implies (\text{Breeze}^t \Leftrightarrow B_{x,y})$$

Logical formulation (dynamic) - cont'd

Some propositions are true at a given time, *e.g.*, to model **current action**

- Forward^t the agent go forward at time t
- Grab^t, Climb^t, TurnLeft^t, TurnRight^t, Shoot^t

We need to model the **transition relation** with **effect axioms**, *e.g.*,

$$L^0_{1,1} \wedge \text{FacingEast}^0 \wedge \text{Forward}^0 \implies (\neg L^1_{1,1} \wedge L^1_{2,1})$$

Note: has to be generalized to each possible time step and each cell.

Frame problem

effect axioms does not state what is remained unchanged!

Logical formulation (dynamic) - cont'd

Two solutions of the frame problem (m actions, n fluent, k changed fluent)

- use **frame axioms** in $\mathcal{O}(mn)$, *e.g.*,

$$\text{Forward}^t \implies (\text{HasArrow}^t \Leftrightarrow \text{HasArrow}^{t+1})$$

- use **successor-state axioms** (reasoning on fluent), in $\mathcal{O}(mk)$, *e.g.*

$$\text{HasArrow}^{t+1} \Leftrightarrow (\text{HasArrow}^t \wedge \neg\text{Shoot}^t)$$

or

$$L_{1,1}^{t+1} \Leftrightarrow (L_{1,1}^t \wedge (\neg\text{Forward}^t \wedge \text{Bump}^{t+1}))\vee (L_{1,2}^t \wedge (\text{FacingSouth}^t \wedge \text{Forward}^t))\vee (L_{2,1}^t \wedge (\text{FacingWest}^t \wedge \text{Forward}^t))$$

Note: to be done for each fluent

Logical formulation (dynamic) - cont'd

Some questions can be also modeled as

$$\mathrm{OK}^t_{x,y} \Leftrightarrow \neg P^t_{x,y} \wedge \neg(W^t_{x,y} \wedge \mathrm{WumpusIsAlive}^t)$$

Wumpus agent

```
function HYBRID-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench,breeze,glitter,bump,scream]
  persistent: KB, a knowledge base, initially the atemporal "wumpus physics"
              t, a counter, initially 0, indicating time
              plan, an action sequence, initially empty

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  TELL the KB the temporal "physics" sentences for time t
  safe ← {[x, y] : ASK(KB, OK^t_{x,y}) = true}
  if ASK(KB, Glitter^t) = true then
    plan ← [Grab] + PLAN-ROUTE(current, {[1,1]}, safe) + [Climb]
  if plan is empty then
    unvisited ← {[x, y] : ASK(KB, L^{t'}_{x,y}) = false for all t' ≤ t}
    plan ← PLAN-ROUTE(current, unvisited ∩ safe, safe)
  if plan is empty and ASK(KB, HaveArrow^t) = true then
    possible_wumpus ← {[x, y] : ASK(KB, ¬ W_{x,y}) = false}
    plan ← PLAN-SHOT(current, possible_wumpus, safe)
  if plan is empty then  // no choice but to take a risk
    not_unsafe ← {[x, y] : ASK(KB, ¬ OK^t_{x,y}) = false}
    plan ← PLAN-ROUTE(current, unvisited ∩ not_unsafe, safe)
  if plan is empty then
    plan ← PLAN-ROUTE(current, {[1, 1]}, safe) + [Climb]
  action ← POP(plan)
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

```
function PLAN-ROUTE(current,goals,allowed) returns an action sequence
  inputs: current, the agent's current position
          goals, a set of squares; try to plan a route to one of them
          allowed, a set of squares that can form part of the route

  problem ← ROUTE-PROBLEM(current, goals,allowed)
  return A*-GRAPH-SEARCH(problem)
```