Lecture 12
Logical Agents, Knowledge Representation & Reasoning: Part II

Rob Gaizauskas

Lecture Outline

- Knowledge-based Agents
- Wumpus World
- Logic
  - Syntax
  - Semantics
  - Model Checking
  - Inference ← Today
- Agents Based on Propositional Logic
- Reading: (Readings that begin with * are **mandatory**)
  - *Russell and Norvig (2010), Chapter 7 "Logical Agents", sections: 7.1-7.4 and 7.7

Inference

- Previous example shows how to derive conclusions given a KB, i.e. how to carry out logical inference
- This approach to inference is called model checking because it enumerates all possible models and checks that α is true in all models in which the KB is true, i.e. that $M(KB) \subseteq M(\alpha)$
- Russell and Norvig analogy: think of set of all consequences of KB as a haystack and α as a needle
  - Entailment is like the needle being in the haystack
  - Inference is like finding it
- If an inference algorithm *i* can be used to derive α from KB write:

$$KB \vdash_i \alpha$$

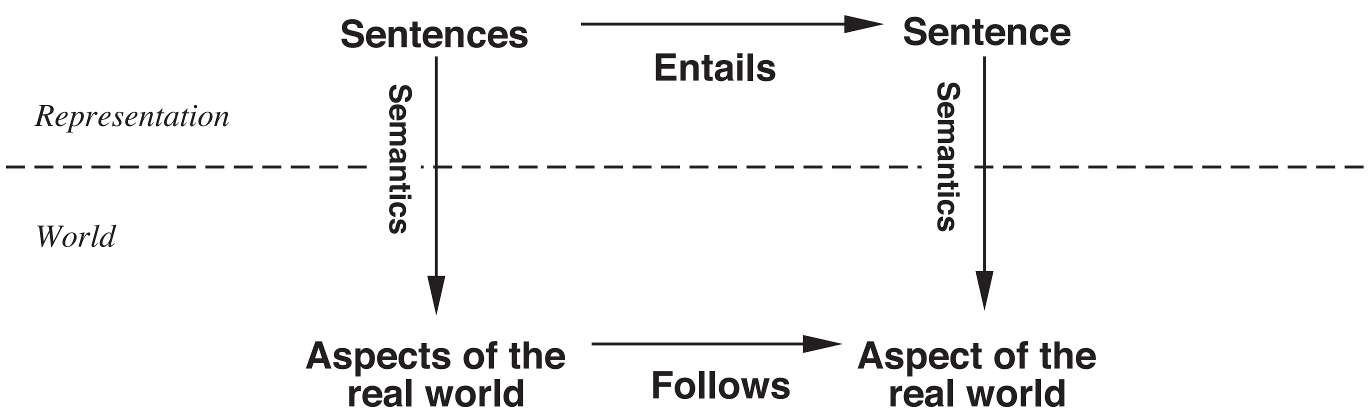
i.e. α is derivable from KB by *i*

Inference: Soundness and Completeness

- An inference procedure that derives only entailed sentences is called sound
- An inference procedure that can derive any entailed sentence is called complete
- In general want inference algorithms to be both sound and complete
  - An unsound algorithm makes things up – "announces the discovery of non-existent needles" (R&N)
  - An incomplete algorithm fails to show that an entailed sentence really is entailed
    - For worlds with a finite number of models, completeness of procedure like model checking is guaranteed
    - For worlds with infinite number of models, completeness is not so straightforward

Inference: In the head vs in the world

- If KB is true in the real world then any sentence derived from it by a sound inference procedure is also true in the real world
- Inference works on "syntax" – internal physical configurations of the agent (bits in registers; brain states)
- But the process corresponds to a real world relationship whereby some aspect of the world is the case as a result of some other aspect of the world being the case



Grounding

- How do we know that the KB is true in the real world?
  - Profound philosophical question
- One answer: agent's sensors create the connection
  - e.g. wumpus agent has a smell sensor – agent creates a suitable sentence whenever there is a smell
    - So whenever the sentence is in the KB it is true in the real world
- Thus, meaning and truth of percept sentences defined by processes of sensing and sentence construction that produce them (i.e. are grounded via sensing)
- General rules, e.g. that wumpuses cause smells in adjacent rooms, are not result of a single percept
  - May be derived from perceptual experience
  - Produced by a sentence construction process called learning
  - generalizations from experience ("inductions") may not always be true

Inference Again

- In computational logic there are two ways to carry out inference:

1. Model checking
   - Checks whether $M(KB) \subseteq M(\alpha)$
   - Naïve approach does brute force enumeration of all models, e.g. via truth table construction, to see if all models satisfying KB also satisfy α
     - $O(2^n)$ complexity where n is number of propositional variables – i.e. exponential
   - More efficient model checking algorithms exist

Inference Again (cont)

2. Theorem proving
   - A proof is a chain of inferences, each justified by a rule which is sound, leading to a desired goal
   - Example of a rule is modus ponens:

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

   - For each rule the truth of the consequent is guaranteed given the truth of the premises/antecedents
   - Finding a proof of a sentence α given a KB amounts to a search for a sequence of sentences such that:
     - Each follows from one or more sentences of the KB or of the preceding sentences in the proof by a rule of inference
     - The last sentence is α

Inference Again (cont)

- Finding a proof may be more efficient than enumerating models because a proof can ignore irrelevant propositions
- Which inference rules are needed to ensure completeness?
  - Several sets have been shown to be sufficient
  - Can in fact get by with only one: the resolution rule (basis of the Prolog logic programming language)

A Propositional Logic Wumpus Agent

- Agent needs to be able to deduce the state of the world given its percept history
- Need to build up a KB that contains:
  - General axioms about how wumpus worlds work
  - Percepts obtained from agent's experience of a particular world
- First focus on deducing current state of the world: where am I? is that square safe? etc.

Axiomatising the Current State of the Wumpus World

- Adopt the following notation:
  - $P_{x,y}$ is true if there is a pit in [x,y]
  - $W_{x,y}$ is true if there is a wumpus in [x,y], dead or alive
  - $B_{x,y}$ is true if the agent perceives a breeze in [x,y]
  - $S_{x,y}$ is true if the agent perceives a stench in [x,y]
- Starting square contains no pit and no wumpus:

  $\neg P_{1,1}$, $\neg W_{1,1}$

Current State of the Wumpus World (2)

- Each square is breezy iff a neighbouring square has a pit. Lots of statements like:

  $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$

  $B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$, etc.

- Each square is smelly iff a neighboring square has a stench. Lots of statements like:

  $S_{1,1} \leftrightarrow (W_{1,2} \vee W_{2,1})$

  $S_{2,1} \leftrightarrow (W_{1,1} \vee P_{2,2} \vee P_{3,1})$, etc.

Current State of the Wumpus World (3)

- There is exactly one wumpus
  - There is at least one wumpus: $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,3} \vee W_{4,4}$
  - There is at most one wumpus (i.e. at least one square is wumpus-free)

    $\neg W_{1,1} \vee \neg W_{1,2}$

    $\neg W_{1,1} \vee \neg W_{1,3}$

    …

    $\neg W_{4,3} \vee \neg W_{4,4}$

Current State of the Wumpus World (4)

- Percept statements need to be indexed by time, since they are different at different times. So:
  - *Stench^t* indicates there is a stench at time *t* (similarly for *Breeze^t* , *Bump^t* , *Glitter^t* and *Scream^t*
- Same idea of associating times with propositions can be used for other aspects of world that change. E.g.:
  - $L^0_{1,1}$: agent is in square [1,1] at time 0
  - FacingEast^0: agent is facing east at time 0
  - HaveArrow^0: agent has an arrow at time 0
  - WumpusAlive^0: wumpus is alive at time 0

Fluents and Atemporal Variables

- Aspects of the world that change with time are called fluents
- Variables associated with non-changing aspects of the world are called atemporal variables
- Can now associate stench and breeze percepts with properties of squares where they are perceived.
  - For any time step t and square [x,y]

$$L_{x,y}^{t} \rightarrow (Breeze^{t} \leftrightarrow B_{x,y})$$

$$L_{x,y}^{t} \rightarrow (Stench^{t} \leftrightarrow S_{x,y})$$

Effect Axioms

- Also need axioms to track fluents such as $L^t_{x,y}$
- Such fluents change as a result of actions by the agent
  - I.e. need a transition model that describes world that results from an agent's actions
- First, need proposition symbols for actions at times
  - Use, e.g., $Forward^0$ to mean agent moves forward at time 0
  - By convention assume
    1. percept at time step happens first
    2. followed by action at time step
    3. followed by transition to next time step

Effect Axioms (2)

- Can now try to describe how world changes by writing effect axioms that describe the effect of an action at next time step
  - If agent is at [1,1] facing east at time 0 and moves Forward then result is that agent is in [2,1] and no longer in [1,1]:

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

  - Need one such sentence for each time, for each of the 16 squares and for each of the 4 orientations.
  - And need similar sentences for the other actions: Grab, Shoot, Climb, TurnRight, TurnLeft

Frame Problem

- Suppose agent moves *Forward* at time 0 and asserts this into the KB
- Given the effect axiom above and the initial assertions about the state at time 0 agent can now deduce it is in [2,1]
  - i.e. ASK(KB, $L^{1}_{2,1}$) = *true*
- However, ASK(KB,$HaveArrow^{1}$) = *false*
  - Agent cannot prove that is still has the arrow
  - Also cannot prove it does NOT have the arrow
- Information has been lost because effect axiom does not state what remains unchanged as a result of an action
- The need to explicitly state what has not changed gives rise to the frame problem

Frame Problem (2)

- One possible solution to the frame problem is to add axioms explicitly asserting all propositions that stay the same for a given action.
- E.g. for each time t:

$$Forward^t \rightarrow (HaveArrrow^t \leftrightarrow HaveArrrow^{t+1})$$
$$Forward^t \rightarrow (WumpusAlive^t \leftrightarrow WumpusAlive^{t+1})$$

  …
- Very inefficient solution
  – In a world with *m* different actions and *n* fluents the set of frame actions will be of size *O(mn)*
- Called the representational frame problem
  – Recall how this was perceived as a major problem for AI in the late 1960s/early 1970s

Frame Problem (3)

- Frame problem significant because really world has very many fluents
  - Most actions change only a small number *k* of these
  - I.e. the world exhibits locality
- So, solving the representational frame problem requires defining a transition model with *O(mk)* axioms instead of *O(mn)*
- Also an inferential frame problem
  - Problem of projecting forward the effects of a *t* step plan of action in time *O(kt)*

Frame Problem (4)

- Solution to frame problem requires changing focus from writing axioms about actions to writing axioms about fluents.
- For each fluent *F* have an axiom that defines truth value of F^{t+1} in terms of
  - fluents at time *t*
  - actions that may have occurred at time *t*
- Truth value of F^{t+1} is set in one of two ways:
  - The action at time *t* causes *F* to be true at *t+1*, or
  - *F* was already true at *t* and action at *t* does not cause it to be false
- Axiom of this type is called a successor-state axiom and has the general form:

$$F^{t+1} \leftrightarrow ActionCausesF^{t} \vee (F^{t} \wedge \neg ActionCausesNotF^{t})$$

Successor State Axioms

- One simple successor-state axiom is for *HaveArrow*
  – Since there is no action for reloading there is no *ActionCausesF^t* part of the axiom and we have only:

$$HaveArrow^{t+1} \leftrightarrow (HaveArrow^{t} \wedge \neg Shoot^{t})$$

- For location, the successor-state axioms are more elaborate.
- For example, $L_{1,1}^{t+1}$ is true if either
  1. The agent moved *Forward* from [1,2] when facing south or from [2,1] when facing west, or
  2. $L_{1,1}^{t}$ was already true and the action did not cause movement (action was not *Forward* or led to bump into wall)

In propositional logic this is written:

$$\begin{aligned} L_{1,1}^{t+1} \leftrightarrow\ & (L_{1,1}^{t} \wedge (\neg Forward^{t} \vee Bump^{t+1})) \\ & \vee (L_{1,2}^{t} \wedge (South^{t} \vee Forward^{l})) \\ & \vee (L_{2,1}^{t} \wedge (West^{t} \vee Forward^{l})) \end{aligned}$$

Current State of the Wumpus World (5)

- Given
  - A complete set of successor state axioms
  - Other axioms introduced above

  Agent can ASK and answer any answerable question about the current state of the world
- For convenience can add axioms

$$\mathrm{OK}^{t}_{x,y} \leftrightarrow \neg P_{x,y} \wedge \neg(W_{x,y} \wedge WumpusAlive^{t})$$

- Now can ask questions like: $ASK(KB, OK^{6}_{2,2}) = true$ to determine whether [2,2] is safe to move into
- With a sound and complete inference algorithm an agent can answer any answerable question about which squares are OK
  - Can can do this in a few milliseconds for small-to-medium wumpus worlds

Current State of the Wumpus World (6)

- The successor-state approach to modelling the effects of actions on fluents solves the representational and inferential frame problems
- However, a problem still remains:
  - For an action to have its intended effect all preconditions the action must be met
    - E.g. *Forward* action moves an agent forward **unless** there is a wall in the way
    - There are many other exceptions that could cause action to fail: agent could trip, malfunction, be abducted by aliens, etc.
  - Problem of specifying all these exceptions is called the qualification problem
- No solution within logic
  - System designers need to use good judgement in deciding how detailed to be in specifying their model and in leaving exceptions out
  - Probability theory offers a means to summarize exceptions without explicitly naming them all

A Hyrid Wumpus World Agent

- Can combine
  - Ability to deduce various aspects of the state of the world
  - Condition-action rules
  - General problem-solving methods

  to produce a hybrid agent for the wumpus world
- Agent program maintains and updates a KB and a plan
- KB:
  - Initial KB contains atemporal axioms – those not depending on *t* such as that a square is breezy if there is a pit nearby
  - At each time step new percept is added + all axioms that depend on *t*, such as the successor-state axioms
  - Then agent uses inference (ASKs) which square are safe and which remain to be visited

A Hyrid Wumpus World Agent (2)

Plan:

- Main part of program constructs a plan based on a decreasing priority of goals
- First, if there is glitter, program constructs a plan to grab the gold, follow a route back to initial location and climb out of cave
- Otherwise, if there is no current plan, program plans a “safe” route to the closest safe square not yet visited
- If there are no safe squares to explore and agent still has an arrow, agent shoots at one of the possible wumpus locations, determined by ASKing where $\neg W_{x,y}$ is false, i.e. where it is not known that there is not a wumpus
- If this fails program looks for a square that is not provably unsafe, i.e. a square for which ASKing if it is not OK returns false
- If there is no such square, mission is impossible and agent retreats to [1,1] and climbs out of cave

Agent Program

Plan

- If there is glitter, construct a plan to grab the gold, follow a route back to initial location and climb out of cave
- Otherwise, if there is no current plan, plan a "safe" route to the closest safe square not yet visited
- If there are no safe squares to explore and agent still has an arrow, shoot at one of the possible wumpus locations
  - determined by ASKing where $\neg W_{x,y}$ is false, i.e. where it is not known that there is not a wumpus
- If this fails look for a square that is not provably unsafe, i.e. a square for which ASKing if it is not OK returns false
- If there is no such square, mission is impossible – retreat to [1,1] and climb out of cave

```
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)
```

Making Plans by Propositional Inference

- The hybrid wumpus world agent
  - Uses inference to determine which squares were safe
  - Uses A* search (a very efficient search algorithm) to make plans
- Can also make plans using logical inference
- Basic idea:

1. Construct a sentence that includes
   a) $Init^0$, a collection of sentences about the initial state
   b) $Transition^1, \ldots, Transition^t$, the successor-state axioms for all possible actions up to some maximum time t
   c) The assertion that the goal is achieved at a time t: $HaveGold^t \wedge ClimbedOut^t$
2. Present the whole sentence to a satisfiability checker (SAT solver). If it finds a satisfying model, the goal is achievable; otherwise it is not.
3. If a model is found, extract from it variables that represent actions and are assigned *true*. These represent a plan to achieve the goals.

Making Plans by Propositional Inference (2)

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

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

- Challenging problem is construction of knowledge base
- Turns out additional axioms are necessary to stop SAT solver assigning values to variables that might help it make a plan. E.g., so far no axioms to preclude the agent:
  - being in two places at once
  - shooting when it does not have an arrow (so, need precondition axioms)
  - carrying out multiple actions simultaneously (need action exclusion axioms)

Making Plans by Propositional Inference (3)

- Modern SAT-solving technologies make the SATPlan approach feasible
  - Can easily find the 11-step solution to the wumpus world instance above
- Thus, can build logical agents that work by
  - Asserting sentences, esp. percept sentences into the KB
  - Performing inference to
    - Determine the current state of the world
    - Create a plan to achieve a goal

Making Plans by Propositional Inference (3)

- Approach has weaknesses
  - When we say “for each time *t*” and “for each square [x,y]” separate propositions with indices for time and squares need to be created and inserted into KB
- For larger worlds, e.g. wumpus world with 100 x 100 board and 1000 time steps, result is a KB with ~10^7 or 10^8 sentences
- This is impractical but also reveals deeper problem:
  - We know that the “physics” of the wumpus world is that same across all squares and times
  - Cannot express this in propositional logic
  - Need a more expressive logical language where we can write “for each time *t*” and “for each square [x,y]” in a more natural way

➡ First-order logic (aka predicate logic)

Summary

- Knowledge-based agents
  - Maintain internal representations of what is the case in the world and how the world changes
  - These internal representations are stored in a knowledge base and typically take the form of sentences in a logical language such as propositional logic or first order logic
  - Reason over these representations to determine the consequences of new percepts or proposed actions or to plan a course of action
- Propositional and 1st order logic agents can and have been built using
  - Either model checking or theorem proving approaches to inference
  - Either hybrid search + reasoning or purely inference-based approaches to planning action sequences

Summary

- Building logical representations of the world requires
  - Modelling aspects of the world that change ("fluents") and those that don't ("atemporal variables")
  - Modelling the effect of agent's actions via effect axioms
  - Modelling what aspects of world change via successor state axioms that describe the behaviour of fluents over time and the effects of actions on fluents
- Challenges for knowledge-based agents include
  - Scaling up to deal with complex worlds with large numbers of states and fluents, extended time sequences, etc.
  - Incorporating probabilities into the reasoning framework

References

Russell, Stuart and Norvig, Peter (2010) Artificial Intelligence: A Modern Introduction (3rd ed). Pearson. Chapter 7.