



Lecture 10 of 42

Logical Agents and Propositional Logic Discussion: Logic in AI

Wednesday, 13 September 2006

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/v9v3>

Course web site: <http://www.kddresearch.org/Courses/Fall-2006/CIS730>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Section 7.5 – 7.7, p. 211 - 232, Russell & Norvig 2nd edition



Lecture Outline

- **Reading for Next Class: Sections 7.5 – 7.7, R&N 2^e**
- **Today: Logical Agents**
 - * Classical knowledge representation
 - * Limitations of the classical symbolic approach
 - * Modern approach: representation, reasoning, learning
 - * “New” aspects: uncertainty, abstraction, classification paradigm
- **Friday: Forward and Backward Chaining in Propositional Logic**
- **Next Week: Start of Material on Logic**
 - * Representation: “a bridge between learning and reasoning” (Koller)
 - * Basis for automated reasoning: theorem proving, other inference
- **Week of 25 Sep 2006: Theorem-Proving Techniques**
 - * Refutation
 - * Resolution and Prolog
- **Week of 02 Oct 2006: Ontology, Epistemology, and KE**





Overview

- **Today's Reading**
 - * Sections 7.1 – 7.4, Russell and Norvig 2e
 - * Recommended references: Nilsson and Genesereth (*Logical Foundations of AI*)
- **Previously: Logical Agents**
 - * Knowledge Bases (KB) and KB agents
 - * Motivating example: Wumpus World
 - * Logic in general
 - * Syntax of propositional calculus
- **Today**
 - * Propositional calculus (concluded)
 - * Normal forms
 - * Production systems
 - * Predicate logic
 - * Introduction to First-Order Logic (FOL): examples, inference rules (sketch)
- **Next Week: First-Order Logic Review, Resolution**



Knowledge Representation (KR) for Intelligent Agent Problems

- **Percepts**
 - * What can agent observe?
 - * What can sensors tell it?
- **Actions**
 - * What actuators does agent have?
 - * In what context are they applicable?
- **Goals**
 - * What are agents goals? Preferences (utilities)?
 - * How does agent evaluate them (check environment, deliberate, etc.)?
- **Environment**
 - * What are “rules of the world”?
 - * How can these be represented, simulated?



Review: Simple Knowledge-Based Agent

```

function KB-AGENT(percept) returns an action
static: KB, a knowledge base
         t, a counter, initially 0, indicating time
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
  
```

The agent must be able to:

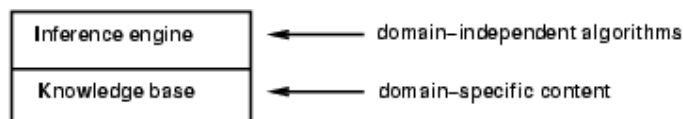
- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Adapted from slides by
S. Russell, UC Berkeley

Figure 6.1 p. 152 R&N



Knowledge bases



- Knowledge base = set of **sentences** in a **formal** language
- **Declarative** approach to building an agent (or other system):
 - * **Tell** it what it needs to know
- Then it can **Ask** itself what to do - answers should follow from KB
- Agents can be viewed at the **knowledge level**
 - i.e., what they know, regardless of how implemented
- Or at the **implementation level**
 - * i.e., data structures in KB and algorithms that manipulate them

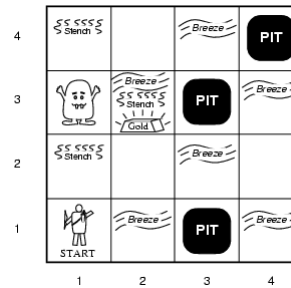
© 2004 S. Russell, UC Berkeley





Wumpus World: PEAS description

- **Performance measure**
 - * gold +1000, death -1000
 - * -1 per step, -10 for using the arrow
- **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
- **Sensors:** Stench, Breeze, Glitter, Bump, Scream
- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot



© 2004 S. Russell, UC Berkeley



Wumpus world: characterization

- **Fully Observable** No – only local perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic** No – sequential at the level of actions
- **Static** Yes – Wumpus and Pits do not move
- **Discrete** Yes
- **Single-agent?** Yes – Wumpus is essentially a natural feature

© 2004 S. Russell, UC Berkeley



Logic in General

Logics are formal languages for representing information such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the “meaning” of sentences; i.e., define truth of a sentence in a world

E.g., the language of arithmetic

$x + 2 \geq y$ is a sentence; $x^2 + y >$ is not a sentence

$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y

$x + 2 \geq y$ is true in a world where $x = 7, y = 1$

$x + 2 \geq y$ is false in a world where $x = 0, y = 6$

Adapted from slides by
S. Russell, UC Berkeley



Entailment

- **Entailment** means that one thing **follows from** another:

$$KB \vdash \alpha$$

- Knowledge base **KB** entails sentence α if and only if α is true in all worlds where **KB** is true

- * E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
- * E.g., $x+y = 4$ entails $4 = x+y$
- * Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

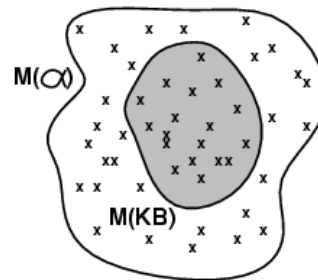
© 2004 S. Russell, UC Berkeley





Models

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
- We say m is a **model** of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
- Then $KB \vdash \alpha$ iff $M(KB) \subseteq M(\alpha)$
 - * E.g. $KB = \text{Giants won and Reds won}$ $\alpha = \text{Giants won}$



© 2004 S. Russell, UC Berkeley



Types of Logic

Logics are characterized by what they commit to as "primitives"

Ontological commitment: what exists—facts? objects? time? beliefs?

Epistemological commitment: what states of knowledge?

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief 0...1
Fuzzy logic	degree of truth	degree of belief 0...1

Adapted from slides by
S. Russell, UC Berkeley

Figure 8.1 p. 244 R&N 2e



Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. $A \quad B \quad C$
 $True \quad True \quad False$

Rules for evaluating truth with respect to a model m :

$\neg S$ is true iff S is false
 $S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true
 $S_1 \vee S_2$ is true iff S_1 is true or S_2 is true
 $S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true
i.e., is false iff S_1 is true and S_2 is false
 $S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true

Adapted from slides by
S. Russell, UC Berkeley



Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences
 - * If S is a sentence, $\neg S$ is a sentence (**negation**)
 - * If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (**conjunction**)
 - * If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (**disjunction**)
 - * If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (**implication**)
 - * If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (**biconditional**)

© 2004 S. Russell, UC Berkeley





Propositional Inference: Enumeration (Model Checking) Method

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that $KB \models \alpha$?

Check all possible models— α must be true wherever KB is true

<i>A</i>	<i>B</i>	<i>C</i>	<i>A</i> \vee <i>C</i>	<i>B</i> \vee \neg <i>C</i>	<i>KB</i>	α
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>
<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>

Adapted from slides by
S. Russell, UC Berkeley



Logical equivalence

- Two sentences are **logically equivalent** iff true in same models: $\alpha \equiv \beta$ iff $\alpha \vdash \beta$ and $\beta \vdash \alpha$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

© 2004 S. Russell, UC Berkeley





Normal Forms: CNF, DNF, Horn

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF—universal)
conjunction of disjunctions of literals
clauses

$$\text{E.g., } (A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$

Disjunctive Normal Form (DNF—universal)
disjunction of conjunctions of literals
terms

$$\text{E.g., } (A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$$

Horn Form (restricted)
conjunction of Horn clauses (clauses with ≤ 1 positive literal)

$$\text{E.g., } (A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$

Often written as set of implications:

$$B \Rightarrow A \text{ and } (C \wedge D) \Rightarrow B$$

Adapted from slides by
S. Russell, UC Berkeley



Validity and Satisfiability

A sentence is valid if it is true in all models
 e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:
 $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model
 e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models
 e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
 $KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable
 i.e., prove α by *reductio ad absurdum*

Adapted from slides by
S. Russell, UC Berkeley





Proof Methods

Proof methods divide into (roughly) two kinds:

Model checking

truth table enumeration (sound and complete for propositional)
heuristic search in model space (sound but incomplete)
e.g., the GSAT algorithm (Ex. 6.15)

Application of inference rules

Legitimate (sound) generation of new sentences from old
Proof = a sequence of inference rule applications

Can use inference rules as operators in a standard search alg.

Adapted from slides by
S. Russell, UC Berkeley



Inference (Sequent) Rules for Propositional Logic

Resolution (for CNF): complete for propositional logic

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Can be used with forward chaining or backward chaining

Adapted from slides by
S. Russell, UC Berkeley





Efficient propositional inference

Two families of efficient algorithms for propositional inference:

Complete backtracking search algorithms

- DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms

`walkSAT` algorithm



DPLL algorithm [1]

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. **Early termination**

A clause is true if any literal is true.

A sentence is false if any clause is false.

2. **Pure symbol heuristic**

Pure symbol: always appears with the same "sign" in all clauses.

e.g., In the three clauses $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A and B are pure, C is impure.

Make a pure symbol literal true.

3. **Unit clause heuristic**

Unit clause: only one literal in the clause

The only literal in a unit clause must be true.

© 2004 S. Russell, UC Berkeley





DPLL algorithm [2]

function DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

inputs: *s*, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of *s*

symbols ← a list of the proposition symbols in *s*

return DPLL(*clauses*, *symbols*, [])

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

if every clause in *clauses* is true in *model* **then return** *true*

if some clause in *clauses* is false in *model* **then return** *false*

P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*-*P*, [*P* = *value*|*model*])

P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*-*P*, [*P* = *value*|*model*])

P ← FIRST(*symbols*); *rest* ← REST(*symbols*)

return DPLL(*clauses*, *rest*, [*P* = *true*|*model*]) **or**

DPLL(*clauses*, *rest*, [*P* = *false*|*model*])

© 2004 S. Russell, UC Berkeley



Inference-based agents in wumpus world

A wumpus-world agent using propositional logic:

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

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

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$

$$W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4}$$

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

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

...

⇒ 64 distinct proposition symbols, 155 sentences

© 2004 S. Russell, UC Berkeley





A propositional logic-Based Wumpus Agent

```

function PL-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench, breeze, glitter]
  static: KB, initially containing the "physics" of the wumpus world
         x, y, orientation, the agent's position (init. [1,1]) and orient. (init. right)
         visited, an array indicating which squares have been visited, initially false
         action, the agent's most recent action, initially null
         plan, an action sequence, initially empty

  update x, y, orientation, visited based on action
  if stench then TELL(KB, Sx,y) else TELL(KB, ¬ Sx,y)
  if breeze then TELL(KB, Bx,y) else TELL(KB, ¬ Bx,y)
  if glitter then action ← grab
  else if plan is nonempty then action ← POP(plan)
  else if for some fringe square [i,j], ASK(KB, (¬ Pi,j ∧ ¬ Wi,j)) is true or
         for some fringe square [i,j], ASK(KB, (Pi,j ∨ Wi,j)) is false then do
    plan ← A*-GRAPH-SEARCH(ROUTE-PB([x,y], orientation, [i,j], visited))
    action ← POP(plan)
  else action ← a randomly chosen move
  return action
  
```

© 2004 S. Russell, UC Berkeley



Limited Expressiveness of propositional logic

- KB contains "physics" sentences for every single square
- For every time t and every location $[x,y]$

$$L_{x,y} \wedge \text{FacingRight}^t \wedge \text{Forward}^t \Rightarrow L_{x+1,y}$$
- Rapid proliferation of clauses

© 2004 S. Russell, UC Berkeley



Summary Points

- **Propositional and First-Order Calculi (Today)**
 - * **Propositional calculus (concluded)**
 - ⇒ Normal forms
 - ⇒ Inference (*aka sequent*) rules
 - * **Production systems**
 - * **Predicate logic without quantifiers**
 - * **Introduction to First-Order Logic (FOL)**
 - ⇒ Examples
 - ⇒ Inference rules (sketch)
- **Friday: Forward and Backward Chaining**
- **Next Week: FOL Intro**
- **Week After: Resolution Theorem Proving, Prolog**



Terminology

- **Logical Frameworks**
 - * **Knowledge Bases (KB)**
 - * **Logic in general: representation languages, syntax, semantics**
 - * **Propositional logic**
 - * **First-order logic (FOL, FOPL)**
 - * **Model theory, domain theory: possible worlds semantics, entailment**
- **Normal Forms**
 - * **Conjunctive Normal Form (CNF)**
 - * **Disjunctive Normal Form (DNF)**
 - * **Horn Form**
- **Proof Theory and Inference Systems**
 - * **Sequent calculi: rules of proof theory**
 - * **Derivability or provability**
 - * **Properties**
 - ⇒ **Soundness (derivability implies entailment)**
 - ⇒ **Completeness (entailment implies derivability)**