Artificial Intelligence
Chapter 7: Logical Agents
Andreas Zell

After the Textbook: Artificial Intelligence, A Modern Approach by Stuart Russel and Peter Norvig (3rd Edition)

7. Logical Agents

• 7.1 Knowledge-Based Agents
• 7.2 The Wumpus World
• 7.3 Logic
• 7.4 Propositional Logic
• 7.5 Propositional Theorem Proving
• 7.6 Effective Propositional Model Checking
• 7.7 Propositional Logic Agents
• 7.8 Summary
7.1 Knowledge-Based Agents

• Logical agents are always definite – each proposition is either true/false or unknown (agnostic)

• Knowledge representation language – a language used to express knowledge about the world
  • Declarative approach – language is designed to be able to easily express knowledge for the world the language is being implemented for
  • Procedural approach – encodes desired behaviors directly in program code

• Sentence – a statement expressing a truth about the world in the knowledge representation language

• Knowledge Base (KB) – a set of sentences describing the world
  • Background knowledge – initial knowledge in KB
  • Knowledge level – we only need to specify what the agent knows and what its goals are in order to specify its behavior
  • Tell(P) – function that adds knowledge P to the KB
  • Ask(P) – function that queries the agent about the truth of P
7.1 Knowledge-Based Agents

- Inference – the process of deriving new sentences from the knowledge base
  - When the agent draws a conclusion from available information, it is guaranteed to be correct if the available information is correct

```c
function KB-Agent( percept) returns an action
   persistent: 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
```

A generic knowledge-based agent

7.2 The Wumpus World

- 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
7.2 The Wumpus World

- Performance measure
  - gold +1000,
  - PIT/wumpus -1000
  - -1 per action,
  - -10 for using the arrow

- Actuators:
  - TurnLeft (90°),
  - TurnRight (90°),
  - Forward,
  - Grab (gold),
  - Shoot (arrow),
  - Climb (at 1,1)

- Sensors:
  - Stench, Breeze, Glitter, Bump, Scream

- 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
7.2 The Wumpus World

First percept at [1,1]
[None, None, None, None, None]
Stench, Breeze, Glitter, Bump, Scream

Percept at [2,1]
[None, Breeze, None, None, None]

Percept at [1,2]
Percept at [2,3]

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.)
7.3 Logic

- **Logics** – formal languages for representing information such that conclusions can be drawn
- **Syntax** – description of a representative language in terms of well-formed sentences of the language
- **Semantics** – defines the “meaning” (truth) of a sentence in the representative language w.r.t. each possible world
- **Model** – the world being described by a KB
- **Satisfaction** – model $m$ satisfies a sentence $\alpha$, if $\alpha$ is true in $m$

---

7.3 Logic

- **Entailment** – the concept that a sentence follows from another sentence:
  - $\alpha \models \beta$ if $\alpha$ is true, then $\beta$ must also be true.
- **Logical inference** – the process of using entailment to derive conclusions
- **Model checking** – enumeration of all possible models to ensure that a sentence $\alpha$ is true in all models in which KB is true
- **$M(\alpha)$** is the set of all models of $\alpha$
7.3 Logic

$KB = \text{wumpus-world rules + observations}$

$\alpha_1 = \"[1,2] is safe\", \ KB \models \alpha_1$, proved by model checking

$KB = \text{wumpus-world rules + observations}$

$\alpha_2 = \"[2,2] is safe\", \ KB \not\models \alpha_2$
7.3 Logic

• If an inference algorithm \( i \) can derive \( \alpha \) from KB we write \( KB \vdash_i \alpha \).

• **Sound (truth-preserving) inference** – an inference algorithm that derives only entailed sentences
  • *if KB is true in the real world, then any sentence \( \alpha \) derived from KB by a sound inference procedure is also true in the real world*

• **Complete inference procedure** – an inference proc. that can derive any sentence that is entailed

• **Grounding** – the connection between logical reasoning processes and the real environment in which the agent exists

---

7.4 Propositional Logic

• **Atomic sentence** – consists of a single propositional symbol, which is *True or False*

• **Complex sentence** – sentence constructed from simpler sentences using parentheses and logical connectives:
  • \( \neg \) (not) – negation
  • \( \land \) (and) – conjunction
  • \( \lor \) (or) – disjunction
  • \( \Rightarrow \) (implies) – implication (premise=>conclusion)
  • \( \Leftrightarrow \) (if and only if) – biconditional

Highest priority

Lowest priority
7.4 Propositional Logic

- **Truth table** – a (simple) representation of a complex sentence by enumerating its truth in terms of the possible values of each of its symbols.

- **Truth table for connectives**:

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P⇒Q</th>
<th>P⇔Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.)

7.4 Propositional Logic

- **Wumpus World Symbols**:
  - P<sub>x,y</sub> is true if there is a pit in [x,y]
  - W<sub>x,y</sub> is true if there is a wumpus in [x,y]
  - B<sub>x,y</sub> is true if there is a breeze in [x,y]
  - S<sub>x,y</sub> is true if there is a stench in [x,y]

- **Sentences R<sub>i</sub>**:
  - No pit in [1,1]
    - R<sub>1</sub>: ¬P<sub>1,1</sub>
  - Pits cause breezes in adjacent squares
    - R<sub>2</sub>: B<sub>1,1</sub> ⇔ (P<sub>1,2</sub> ∨ P<sub>2,1</sub>)
    - R<sub>3</sub>: B<sub>2,1</sub> ⇔ (P<sub>1,1</sub> ∨ P<sub>2,2</sub> ∨ P<sub>3,1</sub>)
  - For first two squares
    - R<sub>4</sub>: ¬B<sub>1,1</sub>
    - R<sub>5</sub>: B<sub>2,1</sub>
7.4 Propositional Logic by Model Checking

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$R_5$</th>
<th>KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Fig. 7.8: Truth Table for Wumpus World KB, consisting of $2^7 = 128$ rows, one each for the different assignments of truth values to the 7 proposition symbols $B_{1,1}, \ldots, P_{3,1}$. KB is true if $R_1$ through $R_5$ are true, which occurs just in 3 rows.

7.4.4 Propositional Model Checking

```
function TT-ENTAILS(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic
    symbols — a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, { })

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY(symbols) then
        if PL-TRUE(KB, model) then return PL-TRUE(α, model)
        else return true // when KB is false, always return true
    else do
        P = FIRST(symbols)
        rest = REST(symbols)
        return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})) and
                TT-CHECK-ALL(KB, α, rest, model ∪ {P = false })
```

Figure 7.8 A truth-table enumeration algorithm for deciding propositional entailment. (TT stands for truth table.) PL-TRUE returns true if a sentence holds within a model. The variable model represents a partial model—an assignment to some of the symbols. The keyword “and” is used here as a logical operation on its two arguments, returning true or false.
7.5 Propositional Theorem Proving

• Knowledge Base can be represented as a conjunction of all its statements since it asserts that all statements are true.

• Every known inference algorithm for propositional logic has a worst-case complexity exponential in the size of the input.

• Logical equivalence – two sentences \( \alpha \) and \( \beta \) are logically equivalent if they are true in the same set of models.

• Validity – a sentence is valid if it is true in all models.

• Valid sentences are also called tautologies – sentences that are necessarily true.

7.5 Propositional Theorem Proving

• Deduction Theorem – For any sentences \( \alpha \) and \( \beta \), \( \alpha \vdash \beta \) if and only if the sentence \( (\alpha \Rightarrow \beta) \) is valid.

• Satisfiability – a sentence is satisfiable if it is true in some model.
  • Determining satisfiability in propositional logic is NP-complete.
  • Proof by contradiction: \( \alpha \vdash \beta \) if and only if the sentence \( \neg(\alpha \Rightarrow \beta) \) or rather \( (\alpha \land \neg \beta) \) is unsatisfiable.

• Inferentially equivalent – two sentences \( \alpha \) and \( \beta \) are inferentially equivalent if the satisfiability of \( \alpha \) implies the satisfiability of \( \beta \) and vice versa.
7.5 Propositional Theorem Proving

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg\alpha \lor \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg\alpha \land \neg\beta) \quad \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]

Fig. 7.11 Standard logical equivalences. The symbols $\alpha$, $\beta$, and $\gamma$ stand for arbitrary sentences of propositional logic.
7.5 Propositional Theorem Proving

Example to prove \( \neg P_{1,2} \) from \( R_1 \) through \( R_5 \):

- Applying biconditional elimination to \( R_2 \) to obtain
  \( R_6: (B_{1,1} \rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)

- Applying And-Elimination to obtain
  \( R_7: ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \)

- Contraposition gives
  \( R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \)

- Modus Ponens with \( R_8 \) and the percept \( \neg B_{1,1} \) gives
  \( R_9: \neg (P_{1,2} \lor P_{2,1}) \)

- De Morgan’s rule gives
  \( R_{10}: \neg P_{1,2} \land \neg P_{2,1} \)

  that is, neither \( P_{12} \) nor \( P_{21} \) contains a pit.

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.)

7.5 Propositional Theorem Proving

Conjunctive Normal Form (CNF) – every sentence of propositional logic is *logically equivalent* to a conjunction of clauses. E.g. Convert \( B_{1,1} \leftrightarrow (P_{1,2} \lor P_{2,1}) \) to CNF:

1. Eliminate \( \leftrightarrow \), replacing \( \alpha \leftrightarrow \beta \) with \((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)\)

   \((B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})\)

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \)

   \( (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \)

3. Move \( \neg \) inwards using de Morgan’s rules and double-negation

   \((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \lor \neg P_{2,1}) \lor B_{1,1})\)

4. Apply distributivity law (\( \lor \) over \( \land \)) and flatten

   \((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})\)

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.)
7.5 Propositional Theorem Proving

Resolution algorithm: Proof by contradiction, i.e., show $KB \land \neg\alpha$ unsatisfiable

Resolution Example from Wumpus World
7.5 Propositional Theorem Proving

- **Definite clause** – disjunction of literals, of which exactly one is positive e.g. \( \neg P_1 \lor \neg P_2 \lor \neg P_3 \lor P_4 \)
- **Horn clause** – a disjunction of literals at most one of which is positive e.g. \( \neg P_1 \lor \neg P_2 \lor \neg P_3 \lor P_4 \)
  - Can be used with forward chaining or backward chaining
  - Deciding entailment is linear in the size of KB
- **Goal clause** – a clause with no positive literals, \( \neg P_1 \lor \neg P_2 \)
- **Forward chaining** – a sound and complete inference algorithm that is essentially Modus Ponens
  - *data-driven reasoning*; reasoning which starts from known data
- **Backward chaining** – *goal-directed reasoning*; reasoning that works backward from goal
  - Often works in much less than linear as it avoids redundant facts
7.5 Propositional Theorem Proving

A set of Horn Clauses

\[
\begin{align*}
P & \rightarrow Q & \neg P \lor Q \\
L \land M & \Rightarrow P & \neg L \lor \neg M \lor P \\
B \land L & \Rightarrow M & \neg B \lor \neg L \lor M \\
A \land P & \Rightarrow L & \neg A \lor \neg P \lor L \\
A \land B & \Rightarrow L & \neg A \lor \neg B \lor L \\
A & & \\
B & &
\end{align*}
\]

And the corresponding AND-OR graph:

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 31

7.5 Propositional Theorem Proving

with Resolution

\[
\begin{align*}
KB = \neg P \lor Q, \neg L \lor \neg M \lor P, \neg B \lor \neg L \lor M, \neg A \lor \neg P \lor L, \neg A \lor \neg B \lor L, A, B
\end{align*}
\]

Question: \( KB \models Q ? \)

\[
KB \models Q \text{ if and only if } KB, \neg Q \models \text{false}
\]

So \( Q, \neg P \lor Q, \neg L \lor \neg M \lor P, \neg B \lor \neg L \lor M, \neg A \lor \neg P \lor L, \neg A \lor \neg B \lor L, A, B \)

\[
\begin{align*}
\neg P & \lor Q, \\
\neg L & \lor \neg M \\
\neg B & \lor \neg L \\
\neg A & \lor \neg B \\
\neg B & \\
\text{false (empty clause)}
\end{align*}
\]

(factoring, elimination of duplicate Literals)

Unit resolution (\( l_i \) are literals):

\[
\begin{align*}
\begin{array}{c}
I_1 \lor I_2 \lor \ldots \lor I_n, \quad \neg l_i \\
I_1 \lor I_2 \lor \ldots \lor I_n, \quad (l_i, \text{deleted})
\end{array}
\end{align*}
\]

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 32
7.5 Propositional Theorem Proving with Resolution

- **Full Resolution:** \[ l_1 \lor l_2 \lor \ldots \lor l_k \lor \neg m_1 \lor \neg m_2 \lor \ldots \lor \neg m_n \]

where the \( l_i \) and \( m_j \) are complementary literals. Multiple copies of a literal are reduced to one (factoring).

Examples:
- \( \neg P \lor Q, \neg L \lor \neg M \lor P \)
- \( \neg B \lor \neg L \lor \neg M \lor P \)
- \( \neg A \lor B, \neg A \lor C \)

\( \neg A \lor B, \neg A \lor C \) cannot be resolved

7.6 Effective Propositional Model Checking

- **Davis-Putnam algorithm (DPLL)** – an algorithm for checking satisfiability based on the fact that satisfiability is commutative. Essentially, it is a DFS method of model checking.

- **Fundamental algorithm:**
  
  DP(\( clauses, symbols, model \))
  - If (all \( clauses \) are true in \( model \)) return true;
  - If (there is a false \( clause \) in \( model \)) return false;
  - \( P \) = next unassigned symbol in \( symbols \);
  - return DP (\( clauses, symbols, model + \{ P / true \} \)) OR DP (\( clauses, symbols, model + \{ P / false \} \));
7.6 Effective Propositional Model Checking

- Heuristics in the Davis-Putnam algorithm:
  - Early termination – short-circuit logical evaluations.
    A clause is true if any literal in it is true.
    A sentence is false if any clause in it is false.
  - Pure symbol heuristic – a symbol that appears with the same sign in all clauses of a sentence (all positive literals or negative ones).
    - Making these literals true can never make a clause false. Hence, pure symbols are fixed respectively.
  - Unit clause heuristic – assignment of true to unit clauses.
    - unit clause – a clause in which all literals but one have been assigned false.
    - unit propagation – assigning one unit clause creates another cascade of forced assignments.

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 propositional symbols in s
return DPLL(clauses, symbols, {})
7.6 Effective Propositional Model Checking

- Tricks to scale up to large SAT problems:
  - Component Analysis (and working on each component separately)
  - Variable and value ordering (choosing the variable that appears most often in remaining clauses)
  - Intelligent backtracking (backing up all the way to the relevant conflict)
  - Random restarts (reduces the variance on the time to solution)
  - Clever indexing (with dynamic indexing structures).

- WalkSAT – a local search algorithm based on the idea of a random walk.
  - Initial assignment is chosen randomly.
  - Repeat until satisfied or “exhausted”.
  - A min-conflicts heuristic (as with CSPs) is used to minimize the number of unsatisfied clauses.
  - A random walk step chooses the symbol to flip.
  - If a satisfying assignment exists, it will be found, eventually.
  - WalkSAT can not guarantee a sentence is unsatisfiable except with high probability.
7.6 Effective Propositional Model Checking

function \textsc{walksat}(\text{clauses}, p, \text{max. flips}) returns a satisfying model or \text{failure}
inputs: clauses, a set of clauses in propositional logic
$p$, the probability of choosing to do a “random walk” move, typically around 0.5
\text{max. flips}, number of flips allowed before giving up

model — a random assignment of \text{true}/\text{false} to the symbols in clauses

for $i = 1$ to \text{max. flips} do
  if model satisfies clauses then return model
  \text{clause} — a randomly selected clause from clauses that is false in model
  with probability $p$ flip the value of a randomly selected symbol in \text{clause}
  else flip whichever symbol in \text{clause} maximizes the number of satisfied clauses

return \text{failure}

Figure 7.15 The \textsc{walksat} algorithm for checking satisfiability by randomly flipping the values of variables. Many versions of the algorithm exist.

7.6 Effective Propositional Model Checking

Hard Satisfiability

- Let $m$ be the number of clauses and $n$ be the number of symbols.
- The ratio $m/n$ is indicative of the difficulty of the problem.
- The probability for satisfiability drops sharply around $m/n = 4.3$.
- underconstrained — relatively small $m/n$ thus making the expected number of satisfying assignments high.
- overconstrained — relatively high $m/n$ thus making the expected number of satisfying assignments low.
- critical point — value of $m/n$ such that the problem is nearly satisfiable and nearly unsatisfiable. Thus, the most difficult cases for satisfiability algorithms

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 39
7.7 Propositional Logic Agents

• **Inference-based agent** – an agent that maintains a knowledge base of propositions and uses the inference procedures described above for reasoning.
  
  • It is beyond the power of propositional logic to efficiently express statements that are true for sets of objects.
  
  • A proliferation of clauses occurs due to the fact that a different set of clauses is needed for each step in time.

• **Circuit-based agent** – a reflex agent in which percepts are inputs to a sequential circuit – a network of gates (logical connectives) and registers (store truth value of a single proposition)
  
  • **dataflow** – at each time step, the inputs are set for that time step and signals propagate through the circuit.
  
  • **delay line** – implements internal state by feeding output of a register back into the register as input at the next time step. The delay is represented as a triangular gate.
  
  • Circuits can only ascribe true/false values to a variable; no unknowns.
    • requires each variable be represented by 2 knowledge propositions; 1 if the variable is known and the other for the value if known.
  
  • **locality** – the property of models in which the truth of each proposition can be determined by a constant number of other propositions.
  
  • **acyclicity** – a circuit such that every cyclical path has a time delay; a requirement for physical implementation.
  
  • Circuit agents have trouble representing interlocking dependencies → incomplete.
7.7 Propositional Logic Agents

- **Tradeoffs:**
  - **Conciseness** – circuit agents do not need separate copies of knowledge at each point in time whereas inference agents do.
  - **Computational Efficiency** – In worst case, inference is exponential in the number of symbols whereas circuit executes linearly in its size.
  - **Completeness** – An inferential agent is complete whereas a complete circuit-based agent becomes exponentially large in the worst case.
  - **Ease of Construction** – Building small, acyclic, not-too-incomplete circuits is relatively hard to building a declarative description.
  - **Hybrid agent** – tries to get the best of both worlds by implementing reflexes with circuit agents and performing inference when needed for more difficult reasoning.

---

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 43

---

```plaintext
7.7 Propositional Logic Agents

Function (Wrap up) to handle perceptual input and return an action
Input: (percept: list [sight,hearing,smell,touch,etc])
Output: an action

```

Function: Plan-Action(current,goal,allowed) returns an action sequence
Input: current, the agent's current position
Output: a sequence of actions the agent can perform to
plan to move to a goal while avoiding obstacles

```

---

Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 44

---