EXERCISES 7

LOGICAL AGENTS

7.1 Knowledge-Based Agents

7.2 The Wumpus World

7.3 Logic

7.4 Propositional Logic: A Very Simple Logic

7.5 Propositional Theorem Proving

7.6 Effective Propositional Model Checking

7.7 Agents Based on Propositional Logic

Exercise 7.WUMT

Suppose the agent has progressed to the point shown in Figure 7.4(a), page 231, having perceived nothing in [1,1], a breeze in [2,1], and a stench in [1,2], and is now concerned with the contents of [1,3], [2,2], and [3,1]. Each of these can contain a pit, and at most one can contain a wumpus. Following the example of Figure 7.5, construct the set of possible worlds. (You should find 32 of t hem.) Mark the worlds in which the KB is true and those in which each of the following sentences is true:

$\alpha_2 =$ "There is no pit in [2,2]."
$\alpha_3 =$ "There is a wumpus in [1,3]."

Hence show that $KB \models \alpha_2$ and $KB \models \alpha_3$.

To save space, we'll show the list of models as a table (Figure S7.1) rather than a collection of diagrams. There are eight possible combinations of pits in the three squares, and four possibilities for the wumpus location (including nowhere).

We can see that $KB \models \alpha_2$ because every line where KB is true also has α_2 true. Similarly for α_3.

Model	KB	α_2	α_3
		true	
$P_{1,3}$		*true*	
$P_{2,2}$			
$P_{3,1}$		*true*	
$P_{1,3}, P_{2,2}$			
$P_{2,2}, P_{3,1}$			
$P_{3,1}, P_{1,3}$		*true*	
$P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{1,3}$		*true*	*true*
$W_{1,3}, P_{1,3}$		*true*	*true*
$W_{1,3}, P_{2,2}$			*true*
$W_{1,3}, P_{3,1}$	*true*	*true*	*true*
$W_{1,3}, P_{1,3}, P_{2,2}$			*true*
$W_{1,3}, P_{2,2}, P_{3,1}$			*true*
$W_{1,3}, P_{3,1}, P_{1,3}$		*true*	*true*
$W_{1,3}, P_{1,3}, P_{3,1}, P_{2,2}$			*true*
$W_{3,1},$		*true*	
$W_{3,1}, P_{1,3}$		*true*	
$W_{3,1}, P_{2,2}$			
$W_{3,1}, P_{3,1}$		*true*	
$W_{3,1}, P_{1,3}, P_{2,2}$			
$W_{3,1}, P_{2,2}, P_{3,1}$			
$W_{3,1}, P_{3,1}, P_{1,3}$		*true*	
$W_{3,1}, P_{1,3}, P_{3,1}, P_{2,2}$			
$W_{2,2}$		*true*	
$W_{2,2}, P_{1,3}$		*true*	
$W_{2,2}, P_{2,2}$			
$W_{2,2}, P_{3,1}$		*true*	
$W_{2,2}, P_{1,3}, P_{2,2}$			
$W_{2,2}, P_{2,2}, P_{3,1}$			
$W_{2,2}, P_{3,1}, P_{1,3}$		*true*	
$W_{2,2}, P_{1,3}, P_{3,1}, P_{2,2}$			

Figure S7.1 A truth table constructed for Ex. 7.2. Propositions not listed as true on a given line are assumed false, and only *true* entries are shown in the table.

Exercise 7.UNIC

(Adapted from Barwise and Etchemendy (1993).) Given the following, can you prove that the unicorn is mythical? How about magical? Horned?

> If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned.

```
**function** PL-TRUE?($\check{s}$, $\check{m}$) **returns** $\check{t}$rue or $\check{f}$alse
   **if** $\check{s}$ = $True$ **then return** $\check{t}$rue
   **else if** $\check{s}$ = $False$ **then return** $\check{f}$alse
   **else if** SYMBOL?($\check{s}$) **then return** LOOKUP($\check{s}$, $\check{m}$)
   **else** branch on the operator of $\check{s}$
      $\neg$: **return not** PL-TRUE?(ARG1($\check{s}$), $\check{m}$)
      $\vee$: **return** PL-TRUE?(ARG1($\check{s}$), $\check{m}$) **or** PL-TRUE?(ARG2($\check{s}$), $\check{m}$)
      $\wedge$: **return** PL-TRUE?(ARG1($\check{s}$), $\check{m}$) **and** PL-TRUE?(ARG2($\check{s}$), $\check{m}$)
      $\Rightarrow$: (**not** PL-TRUE?(ARG1($\check{s}$), $\check{m}$)) **or** PL-TRUE?(ARG2($\check{s}$), $\check{m}$)
      $\Leftrightarrow$: PL-TRUE?(ARG1($\check{s}$), $\check{m}$) **iff** PL-TRUE?(ARG2($\check{s}$), $\check{m}$)
```

Figure S7.2 Pseudocode for evaluating the truth of a sentence wrt a model.

The unicorn is magical if it is horned.

As human reasoners, we can see from the first two statements, that if it is mythical, then it is immortal; otherwise it is a mammal. So it must be either immortal or a mammal, and thus horned. That means it is also magical. However, we can't deduce anything about whether it is mythical. To proide a formal answer, we can enumerate the possible worlds ($2^5 = 32$ of them with 5 proposition symbols), mark those in which all the assertions are true, and see which conclusions hold in all of those. Or, we can let the machine do the work—in this case, the Lisp code for propositional reasoning:

```
> (setf kb (make-prop-kb))
#S(PROP-KB SENTENCE (AND))
> (tell kb "Mythical => Immortal")
T
> (tell kb "~Mythical => ~Immortal ^ Mammal")
T
> (tell kb "Immortal | Mammal => Horned")
T
> (tell kb "Horned => Magical")
T
> (ask kb "Mythical")
NIL
> (ask kb "~Mythical")
NIL
> (ask kb "Magical")
T
> (ask kb "Horned")
T
```

Exercise 7.TRUV

Consider the problem of deciding whether a propositional logic sentence is true in a given model.

a. Write a recursive algorithm PL-TRUE?(s, m) that returns true if and only if the sentence s is true in the model m (where m assigns a truth value for every symbol in s). The algorithm should run in time linear in the size of the sentence. (Alternatively, use a version of this function from the online code repository.)

b. Give three examples of sentences that can be determined to be true or false in a *partial* model that does not specify a truth value for some of the symbols.

c. Show that the truth value (if any) of a sentence in a partial model cannot be determined efficiently in general.

d. Modify your PL-TRUE? algorithm so that it can sometimes judge truth from partial models, while retaining its recursive structure and linear run time. Give three examples of sentences whose truth in a partial model is *not* detected by your algorithm.

e. Investigate whether the modified algorithm makes TT-ENTAILS? more efficient.

a. See Figure S7.2. We assume the language has built-in Boolean operators **not**, **and**, **or**, **iff**.

b. The question is somewhat ambiguous: we can interpret "in a *partial* model" to mean in *all* such models or *some* such models. For the former interpretation, the sentences $False \wedge P$, $True \vee \neg P$, and $P \wedge \neg P$ can all be determined to be true or false in any partial model. For the latter interpretation, we can in addition have sentences such as $A \wedge P$ which is false in the partial model $\{A = false\}$.

c. A general algorithm for partial models must handle the empty partial model, with no assignments. In that case, the algorithm must determine validity and unsatisfiability, which are co-NP-complete and NP-complete respectively.

d. It helps if **and** and **or** evaluate their arguments in sequence, terminating on false or true arguments, respectively. In that case, the algorithm already has the desired properties: in the partial model where P is true and Q is unknown, $P \vee Q$ returns true, and $\neg P \wedge Q$ returns false. But the truth values of $Q \vee \neg Q$, $Q \vee True$, and $Q \wedge \neg Q$ are not detected.

e. Early termination in Boolean operators will provide a very substantial speedup. In most languages, the Boolean operators already have the desired property, so you would have to write special "dumb" versions and observe a slow-down.

Exercise 7.TFMO

Which of the following are correct?

a. $False \models True$.

b. $True \models False$.

c. $(A \wedge B) \models (A \Leftrightarrow B)$.

d. $A \Leftrightarrow B \models A \vee B$.

e. $A \Leftrightarrow B \models \neg A \lor B$.
f. $(A \land B) \Rightarrow C \models (A \Rightarrow C) \lor (B \Rightarrow C)$.
g. $(C \lor (\neg A \land \neg B)) \equiv ((A \Rightarrow C) \land (B \Rightarrow C))$.
h. $(A \lor B) \land (\neg C \lor \neg D \lor E) \models (A \lor B)$.
i. $(A \lor B) \land (\neg C \lor \neg D \lor E) \models (A \lor B) \land (\neg D \lor E)$.
j. $(A \lor B) \land \neg(A \Rightarrow B)$ is satisfiable.
k. $(A \Leftrightarrow B) \land (\neg A \lor B)$ is satisfiable.
l. $(A \Leftrightarrow B) \Leftrightarrow C$ has the same number of models as $(A \Leftrightarrow B)$ for any fixed set of proposition symbols that includes A, B, C.

In all cases, the question can be resolved easily by referring to the definition of entailment.

a. *False* $\models$ *True* is true because *False* has no models and hence entails every sentence AND because *True* is true in all models and hence is entailed by every sentence.

b. *True* $\models$ *False* is false.

c. $(A \land B) \models (A \Leftrightarrow B)$ is true because the left-hand side has exactly one model that is one of the two models of the right-hand side.

d. $A \Leftrightarrow B \models A \lor B$ is false because one of the models of $A \Leftrightarrow B$ has both A and B false, which does not satisfy $A \lor B$.

e. $A \Leftrightarrow B \models \neg A \lor B$ is true because the RHS is $A \Rightarrow B$, one of the conjuncts in the definition of $A \Leftrightarrow B$.

f. $(A \land B) \Rightarrow C \models (A \Rightarrow C) \lor (B \Rightarrow C)$ is true because the RHS is false only when both disjuncts are false, i.e., when A and B are true and C is false, in which case the LHS is also false. This may seem counterintuitive, and would not hold if $\Rightarrow$ is interpreted as "causes."

g. $(C \lor (\neg A \land \neg B)) \equiv ((A \Rightarrow C) \land (B \Rightarrow C))$ is true; proof by truth table enumeration, or by application of distributivity (Fig 7.11).

h. $(A \lor B) \land (\neg C \lor \neg D \lor E) \models (A \lor B)$ is true; removing a conjunct only allows more models.

i. $(A \lor B) \land (\neg C \lor \neg D \lor E) \models (A \lor B) \land (\neg D \lor E)$ is false; removing a disjunct allows fewer models.

j. $(A \lor B) \land \neg(A \Rightarrow B)$ *is* satisfiable; model has A and $\neg B$.

k. $(A \Leftrightarrow B) \land (\neg A \lor B)$ *is* satisfiable; RHS is entailed by LHS so models are those of $A \Leftrightarrow B$.

l. $(A \Leftrightarrow B) \Leftrightarrow C$ does have the same number of models as $(A \Leftrightarrow B)$; half the models of $(A \Leftrightarrow B)$ satisfy $(A \Leftrightarrow B) \Leftrightarrow C$, as do half the non-models, and there are the same numbers of models and non-models.

Exercise 7.DEDU

Prove each of the following assertions:

a. α is valid if and only if $\mathit{True} \models \alpha$.
b. For any α, $\mathit{False} \models \alpha$.
c. $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.
d. $\alpha \equiv \beta$ if and only if the sentence $(\alpha \Leftrightarrow \beta)$ is valid.
e. $\alpha \models \beta$ if and only if the sentence $(\alpha \land \neg\beta)$ is unsatisfiable.

Remember, $\alpha \models \beta$ iff in every model in which α is true, β is also true. Therefore,

a. α is valid if and only if $\mathit{True} \models \alpha$.
Forward: If $alpha$ is *valid* it is true in all models, hence it is true in all models of True.
Backward: if $\mathit{True} \models \alpha$ then α must be true in all models of True, i.e., in all models, hence α must be valid.

b. For any α, $\mathit{False} \models \alpha$.
$False$ doesn't hold in any model, so α trivially holds in every model of $False$.

c. $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.
Both sides are equivalent to the assertion that there is no model in which α is true and β is false, i.e., no model in which $\alpha \Rightarrow \beta$ is false.

d. $\alpha \equiv \beta$ if and only if the sentence $(\alpha \Leftrightarrow \beta)$ is valid.
Both sides are equivalent to the assertion that α and β have the same truth value in every model.

e. $\alpha \models \beta$ if and only if the sentence $(\alpha \land \neg\beta)$ is unsatisfiable.
As in **c**, both sides are equivalent to the assertion that there is no model in which α is true and β is false.

Exercise 7.PRVM

Prove, or find a counterexample to, each of the following assertions:

a. If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \land \beta) \models \gamma$
b. If $(\alpha \land \beta) \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).
c. If $\alpha \models (\beta \lor \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).

a. If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \land \beta) \models \gamma$.
True. This follows from monotonicity.

b. If $(\alpha \land \beta) \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).
False. Consider Consider $\alpha \equiv A$, $\beta \equiv B$, $\gamma \equiv (A \land B)$.

c. If $\alpha \models (\beta \lor \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).
False. Consider $\beta \equiv A$, $\gamma \equiv \neg A$.

Exercise 7.ABCD

Consider a vocabulary with only four propositions, A, B, C, and D. How many models are there for the following sentences?

a. $B \vee C$.
b. $\neg A \vee \neg B \vee \neg C \vee \neg D$.
c. $(A \Rightarrow B) \wedge A \wedge \neg B \wedge C \wedge D$.

These can be computed by counting the rows in a truth table that come out true, but each has some simple property that allows a short-cut:

a. Sentence is false only if B and C are false, which occurs in 4 cases for A and D, leaving 12.
b. Sentence is false only if A, B, C, and D are false, which occurs in 1 case, leaving 15.
c. The last four conjuncts specify a model in which the first conjunct is false, so 0.

Exercise 7.BCON
We have defined four binary logical connectives.

a. Are there any others that might be useful?
b. How many binary connectives can there be?
c. Why are some of them not very useful?

A binary logical connective is defined by a truth table with 4 rows. Each of the four rows may be true or false, so there are $2^4 = 16$ possible truth tables, and thus 16 possible connectives. Six of these are trivial ones that ignore one or both inputs; they correspond to $True$, $False$, P, Q, $\neg P$ and $\neg Q$. Four of them we have already studied: $\wedge, \vee, \Rightarrow, \Leftrightarrow$. The remaining six are potentially useful. One of them is reverse implication ($\Leftarrow$ instead of $\Rightarrow$), and the other five are the negations of $\wedge, \vee, \Leftrightarrow, \Rightarrow$ and $\Leftarrow$. The first three of these are sometimes called *nand*, *nor*, and *xor*.

Exercise 7.LGEQ
Using a method of your choice, verify each of the equivalences in Figure 7.11 (page 241).

We use the truth table code in Lisp in the directory `logic/prop.lisp` to show each sentence is valid. We substitute `P, Q, R` for α, β, γ because of the lack of Greek letters in ASCII. To save space in this manual, we only show the first four truth tables:

```
> (truth-table "P ^ Q <=> Q ^ P")
-----------------------------------------
 P  Q  P ^ Q  Q ^ P  (P ^ Q) <=> (Q ^ P)
-----------------------------------------
 F  F    F      F             \(true\)
 T  F    F      F              T
 F  T    F      F              T
 T  T    T      T            T
-----------------------------------------
NIL

> (truth-table "P | Q <=> Q | P")
-----------------------------------------
 P  Q  P | Q  Q | P  (P | Q) <=> (Q | P)
-----------------------------------------
 F  F    F      F            T
 T  F    T      T            T
 F  T    T      T            T
 T  T    T      T            T
-----------------------------------------
NIL

> (truth-table "P ^ (Q ^ R) <=> (P ^ Q) ^ R")
-----------------------------------------------------------------------
 P  Q  R  Q ^ R  P ^ (Q ^ R)  P ^ Q ^ R  (P ^ (Q ^ R)) <=> (P ^ Q ^ R)
-----------------------------------------------------------------------
 F  F  F    F         F           F                    T
 T  F  F    F         F           F                    T
 F  T  F    F         F           F                    T
 T  T  F    F         F           F                    T
 F  F  T    F         F           F                    T
 T  F  T    F         F           F                    T
 F  T  T    T         F           F                    T
 T  T  T    T         T           T                    T
-----------------------------------------------------------------------
NIL

> (truth-table "P | (Q | R) <=> (P | Q) | R")
-----------------------------------------------------------------------
 P  Q  R  Q | R  P | (Q | R)  P | Q | R  (P | (Q | R)) <=> (P | Q | R)
-----------------------------------------------------------------------
 F  F  F    F         F           F                    T
 T  F  F    F         T           T                    T
 F  T  F    T         T           T                    T
 T  T  F    T         T           T                    T
 F  F  T    T         T           T                    T
 T  F  T    T         T           T                    T
 F  T  T    T         T           T                    T
 T  T  T    T         T           T                    T
-----------------------------------------------------------------------
NIL
```

For the remaining sentences, we just show that they are valid according to the `validity` function:

```
> (validity "~~P <=> P")
VALID
> (validity "P => Q <=> ~Q => ~P")
VALID
> (validity "P => Q <=> ~P | Q")
VALID
> (validity "(P <=> Q) <=> (P => Q) ^ (Q => P)")
VALID
```

```
> (validity "~(P ^ Q) <=> ~P | ~Q")
VALID
> (validity "~(P | Q) <=> ~P ^ ~Q")
VALID
> (validity "P ^ (Q | R) <=> (P ^ Q) | (P ^ R)")
VALID
> (validity "P | (Q ^ R) <=> (P | Q) ^ (P | R)")
VALID
```

Exercise 7.VUNN

Decide whether each of the following sentences is valid, unsatisfiable, or neither. Verify your decisions using truth tables or the equivalence rules of Figure 7.11 (page 241).

a. $\mathit{Smoke} \Rightarrow \mathit{Smoke}$
b. $\mathit{Smoke} \Rightarrow \mathit{Fire}$
c. $(\mathit{Smoke} \Rightarrow \mathit{Fire}) \Rightarrow (\neg \mathit{Smoke} \Rightarrow \neg \mathit{Fire})$
d. $\mathit{Smoke} \lor \mathit{Fire} \lor \neg \mathit{Fire}$
e. $((\mathit{Smoke} \land \mathit{Heat}) \Rightarrow \mathit{Fire}) \Leftrightarrow ((\mathit{Smoke} \Rightarrow \mathit{Fire}) \lor (\mathit{Heat} \Rightarrow \mathit{Fire}))$
f. $(\mathit{Smoke} \Rightarrow \mathit{Fire}) \Rightarrow ((\mathit{Smoke} \land \mathit{Heat}) \Rightarrow \mathit{Fire})$
g. $\mathit{Big} \lor \mathit{Dumb} \lor (\mathit{Big} \Rightarrow \mathit{Dumb})$

a. Valid.
b. Neither.
c. Neither.
d. Valid.
e. Valid.
f. Valid.
g. Valid.

Exercise 7.LEPW

Any propositional logic sentence is logically equivalent to the assertion that each possible world in which it would be false is not the case. From this observation, prove that any sentence can be written in CNF.

Each possible world can be written as a conjunction of literals, e.g. $(A \land B \land \neg C)$. Asserting that a possible world is not the case can be written by negating that, e.g. $\neg(A \land B \land \neg C)$, which can be rewritten as $(\neg A \lor \neg B \lor C)$. This is the form of a clause; a conjunction of these clauses is a CNF sentence, and can list the negations of all the possible worlds that would make the sentence false.

Exercise 7.RESO

Use resolution to prove the sentence $\neg A \wedge \neg B$ from the clauses in Exercise CONVERT-CLAUSAL-EXERCISE.

To prove the conjunction, it suffices to prove each literal separately. To prove $\neg B$, add the negated goal S7: B.

- Resolve S7 with S5, giving S8: F.
- Resolve S7 with S6, giving S9: C.
- Resolve S8 with S3, giving S10: $(\neg C \vee \neg B)$.
- Resolve S9 with S10, giving S11: $\neg B$.
- Resolve S7 with S11 giving the empty clause.

To prove $\neg A$, add the negated goal S7: A.

- Resolve S7 with the first clause of S1, giving S8: $(B \vee E)$.
- Resolve S8 with S4, giving S9: B.
- Proceed as above to derive the empty clause.

(Note: Early printings of the international edition had a typo, asking for a proof of $\neg A \wedge \neg B$ rather than $\neg A \wedge \neg C$.) To prove the conjunction, it suffices to prove each literal separately. To prove $\neg C$, add the negated goal S7: C.

- Resolve S7 with S5, giving S8: F.
- Resolve S7 with S6, giving S9: B.
- Resolve S8 with S3, giving S10: $(\neg B \vee \neg C)$.
- Resolve S9 with S10, giving S11: $\neg C$.
- Resolve S7 with S11 giving the empty clause.

To prove $\neg A$, add the negated goal S7: A.

- Resolve S7 with the first clause of S1, giving S8: $(C \vee E)$.
- Resolve S8 with S4, giving S9: C.
- Proceed as above to derive the empty clause.

Exercise 7.CLIM

This exercise looks into the relationship between clauses and implication sentences.

a. Show that the clause $(\neg P_1 \vee \cdots \vee \neg P_m \vee Q)$ is logically equivalent to the implication sentence $(P_1 \wedge \cdots \wedge P_m) \Rightarrow Q$.

b. Show that every clause (regardless of the number of positive literals) can be written in the form $(P_1 \wedge \cdots \wedge P_m) \Rightarrow (Q_1 \vee \cdots \vee Q_n)$, where the Ps and Qs are proposition symbols. A knowledge base consisting of such sentences is in **implicative normal form** or **Kowalski form** (Kowalski, 1979).

c. Write down the full resolution rule for sentences in implicative normal form.

a. $P \Rightarrow Q$ is equivalent to $\neg P \lor Q$ by implication elimination (Figure 7.11), and $\neg(P_1 \land \cdots \land P_m)$ is equivalent to $(\neg P_1 \lor \cdots \lor \neg P_m)$ by de Morgan's rule, so $(\neg P_1 \lor \cdots \lor \neg P_m \lor Q)$ is equivalent to $(P_1 \land \cdots \land P_m) \Rightarrow Q$.

b. A clause can have positive and negative literals; let the negative literals have the form $\neg P_1, \ldots, \neg P_m$ and let the positive literals have the form $Q_1, \ldots, Q_n$, where the P_is and Q_js are symbols. Then the clause can be written as $(\neg P_1 \lor \cdots \lor \neg P_m \lor Q_1 \lor \cdots \lor Q_n)$. By the previous argument, with $Q = Q_1 \lor \cdots \lor Q_n$, it is immediate that the clause is equivalent to

$$(P_1 \land \cdots \land P_m) \Rightarrow Q_1 \lor \cdots \lor Q_n\ .$$

c. For atoms p_i, q_i, r_i, s_i where $p_j = q_k$:

$$\frac{\begin{array}{c} p_1 \land \ldots\ p_j\ \ldots \land p_{n_1} \Rightarrow r_1 \lor \ldots r_{n_2} \\ s_1 \land \ldots \land s_{n_3} \Rightarrow q_1 \lor \ldots\ q_k\ \ldots \lor q_{n_4} \end{array}}{p_1 \land \ldots p_{j-1} \land p_{j+1} \land p_{n_1} \land s_1 \land \ldots s_{n_3} \Rightarrow r_1 \lor \ldots r_{n_2} \lor q_1 \lor \ldots\ q_{k-1} \lor q_{k+1} \lor \ldots \lor q_{n_4}}$$

Exercise 7.RADI

According to some political pundits, a person who is radical (R) is electable (E) if he/she is conservative (C), but otherwise is not electable.

a. Which of the following are correct representations of this assertion?

(i) $(R \land E) \iff C$

(ii) $R \Rightarrow (E \iff C)$

(iii) $R \Rightarrow ((C \Rightarrow E) \lor \neg E)$

b. Which of the sentences in (a) can be expressed in Horn form?

a. Correct representations of "a person who is radical is electable if he/she is conservative, but otherwise is not electable":

(i) $(R \land E) \iff C$
No; this sentence asserts, among other things, that all conservatives are radical, which is not what was stated.

(ii) $R \Rightarrow (E \iff C)$
Yes, this says that if a person is a radical then they are electable if and only if they are conservative.

(iii) $R \Rightarrow ((C \Rightarrow E) \lor \neg E)$
No, this is equivalent to $\neg R \lor \neg C \lor E \lor \neg E$ which is a tautology, true under any assignment.

b. Horn form:

(i) Yes:

$$\begin{aligned}(R \land E) \iff C &\equiv ((R \land E) \Rightarrow C) \land (C \Rightarrow (R \land E)) \\ &\equiv ((R \land E) \Rightarrow C) \land (C \Rightarrow R) \land (C \Rightarrow E)\end{aligned}$$

(ii) Yes:

$$\begin{aligned} R \Rightarrow (E \iff C) &\equiv R \Rightarrow ((E \Rightarrow C) \land (C \Rightarrow E)) \\ &\equiv \neg R \lor ((\neg E \lor C) \land (\neg C \lor E)) \\ &\equiv (\neg R \lor \neg E \lor C) \land (\neg R \lor \neg C \lor E) \end{aligned}$$

(iii) Yes, e.g., $True \Rightarrow True$.

Exercise 7.SATC

This question considers representing satisfiability (SAT) problems as CSPs.

a. Draw the constraint graph corresponding to the SAT problem

$$(\neg X_1 \lor X_2) \land (\neg X_2 \lor X_3) \land \ldots \land (\neg X_{n-1} \lor X_n)$$

for the particular case $n = 5$.

b. How many solutions are there for this general SAT problem as a function of n?

c. Suppose we apply BACKTRACKING-SEARCH (page 210) to find *all* solutions to a SAT CSP of the type given in (a). (To find *all* solutions to a CSP, we simply modify the basic algorithm so it continues searching after each solution is found.) Assume that variables are ordered $X_1, \ldots, X_n$ and *false* is ordered before *true*. How much time will the algorithm take to terminate? (Write an $O(\cdot)$ expression as a function of n.)

d. We know that SAT problems in Horn form can be solved in linear time by forward chaining (unit propagation). We also know that every tree-structured binary CSP with discrete, finite domains can be solved in time linear in the number of variables (Section 6.5). Are these two facts connected? Discuss.

a. The graph is simply a connected chain of 5 nodes, one per variable.

b. $n + 1$ solutions. Once any X_i is true, all subsequent X_js must be true. Hence the solutions are i falses followed by $n - i$ trues, for $i = 0, \ldots, n$.

c. The complexity is $O(n^2)$. This is somewhat tricky. Consider what part of the complete binary tree is explored by the search. The algorithm must follow all solution sequences, which themselves cover a quadratic-sized portion of the tree. Failing branches are all those trying a $false$ after the preceding variable is assigned $true$. Such conflicts are detected immediately, so they do not change the quadratic cost.

d. These facts are not obviously connected. Horn-form logical inference problems need not have tree-structured constraint graphs; the linear complexity comes from the nature of the constraint (implication) not the structure of the problem.

Exercise 7.ESAT

Explain why every nonempty propositional clause, by itself, is satisfiable. Prove rigorously that every set of five 3-SAT clauses is satisfiable, provided that each clause mentions exactly three distinct variables. What is the smallest set of such clauses that is unsatisfiable? Construct such a set.

A clause is a disjunction of literals, and its models are the *union* of the sets of models of each literal; and each literal satisfies half the possible models. (Note that $False$ is unsatisfiable, but it is really another name for the empty clause.) A 3-SAT clause with three distinct variables rules out exactly 1/8 of all possible models, so five clauses can rule out no more than 5/8 of the models. Eight clauses are needed to rule out all models. Suppose we have variables A, B, C. There are eight models, and we write one clause to rule out each model. For example, the model $\{A = false, B = false, C = false\}$ is ruled out by the clause $(\neg A \vee \neg B \vee \neg C)$.

Exercise 7.TCNF

A propositional *2-CNF* expression is a conjunction of clauses, each containing *exactly 2* literals, e.g.,

$$(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee D) \wedge (\neg C \vee G) \wedge (\neg D \vee G)\ .$$

a. Prove using resolution that the above sentence entails G.

b. Two clauses are *semantically distinct* if they are not logically equivalent. How many semantically distinct 2-CNF clauses can be constructed from n proposition symbols?

c. Using your answer to (b), prove that propositional resolution always terminates in time polynomial in n given a 2-CNF sentence containing no more than n distinct symbols.

d. Explain why your argument in (c) does not apply to 3-CNF.

a. The negated goal is $\neg G$. Resolve witht he last two clauses to produce $\neg C$ and $\neg D$. Resolve with the second and third clauses to produce $\neg A$ and $\neg B$. Resolve these successively against the first clause to produce the empty clause.

b. This can be answered with or without $True$ and $False$ symbols; we'll omit them for simplicity. First, each 2-CNF clause has two places to put literals. There are $2n$ distinct literals, so there are $(2n)^2$ syntactically distinct clauses. Now, many of these clauses are semantically identical. Let us handle them in groups. There are $C(2n, 2) = (2n)(2n - 1)/2 = 2n^2 - n$ clauses with two different literals, if we ignore ordering. All these clauses are semantically distinct except those that are equivalent to $True$ (e.g., $(A \vee \neg A)$), of which there are n, so that makes $2n^2 - 2n + 1$ clauses with distinct literals. There are $2n$ clauses with repeated literals, all distinct. So there are $2n^2 + 1$ distinct clauses in all.

c. Resolving two 2-CNF clauses cannot increase the clause size; therefore, resolution can generate only $O(n^2)$ distinct clauses before it must terminate.

d. First, note that the number of 3-CNF clauses is $O(n^3)$, so we cannot argue for nonpolynomial complexity on the basis of the number of different clauses! The key observation is that resolving two 3-CNF clauses can *increase* the clause size to 4, and so on, so clause size can grow to $O(n)$, giving $O(2^n)$ possible clauses.

Exercise 7.PRPC

Prove each of the following assertions:

a. Every pair of propositional clauses either has no resolvents, or all their resolvents are logically equivalent.

b. There is no clause that, when resolved with itself, yields (after factoring) the clause $(\neg P \lor \neg Q)$.

c. If a propositional clause C can be resolved with a copy of itself, it must be logically equivalent to True.

a. If the clauses have no complementary literals, they have no resolvents. If they have one pair of complementary literals, they have one resolvent, which is logically equivalent to itself. If they have more than one pair, then one pair resolves away and the other pair appears in the resolvent as $(\ldots A \lor \neg A \ldots)$ which renders the resolvent logically equivalent to $True$.

b. The original clause must include both $\neg P$ and $\neg Q$), for them to appear in the resolvent. There must be a third literal that was resolved away. It must be either P or Q, since any other literal would not be complementary to any of the literals in the clause. If it were P, then one copy of P would be in the resolvent; the same goes for Q.

c. Such a clause must contain a literal and its complement, and their disjunction is a tautology.

Exercise 7.FOOD

Consider the following sentence:

$$[(\mathit{Food} \Rightarrow \mathit{Party}) \lor (\mathit{Drinks} \Rightarrow \mathit{Party})] \Rightarrow [(\mathit{Food} \land \mathit{Drinks}) \Rightarrow \mathit{Party}]\,.$$

a. Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.

b. Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).

c. Prove your answer to (a) using resolution.

a. A simple truth table has eight rows, and shows that the sentence is true for all models and hence valid.

b. For the left-hand side we have:

$$\begin{array}{l}(Food \Rightarrow Party) \lor (Drinks \Rightarrow Party)\\ (\neg Food \lor Party) \lor (\neg Drinks \lor Party)\\ (\neg Food \lor Party \lor \neg Drinks \lor Party)\\ (\neg Food \lor \neg Drinks \lor Party)\end{array}$$

and for the right-hand side we have

$$\begin{array}{l}(Food \land Drinks) \Rightarrow Party\\ \neg(Food \land Drinks) \lor Party\\ (\neg Food \lor \neg Drinks) \lor Party\\ (\neg Food \lor \neg Drinks \lor Party)\end{array}$$

The two sides are identical in CNF, and hence the original sentence is of the form $P \Rightarrow P$, which is valid for any P.

c. To prove that a sentence is valid, prove that its negation is unsatisfiable. I.e., negate it, convert to CNF, use resolution to prove a contradiction. We can use the above CNF result for the LHS.

$$\begin{array}{l}\neg[[(Food \Rightarrow Party) \lor (Drinks \Rightarrow Party)] \Rightarrow [(Food \land Drinks) \Rightarrow Party]]\\ [(Food \Rightarrow Party) \lor (Drinks \Rightarrow Party)] \land \neg[(Food \land Drinks) \Rightarrow Party]\\ (\neg Food \lor \neg Drinks \lor Party) \land Food \land Drinks \land \neg Party\end{array}$$

Each of the three unit clauses resolves in turn against the first clause, leaving an empty clause.

Exercise 7.DNFX

A sentence is in **disjunctive normal form** (DNF) if it is the disjunction of conjunctions of literals. For example, the sentence $(A \land B \land \neg C) \lor (\neg A \land C) \lor (B \land \neg C)$ is in DNF.

a. Any propositional logic sentence is logically equivalent to the assertion that some possible world in which it would be true is in fact the case. From this observation, prove that any sentence can be written in DNF.

b. Construct an algorithm that converts any sentence in propositional logic into DNF. (*Hint*: The algorithm is similar to the algorithm for conversion to CNF given in Section 7.5.2.)

c. Construct a simple algorithm that takes as input a sentence in DNF and returns a satisfying assignment if one exists, or reports that no satisfying assignment exists.

d. Apply the algorithms in (b) and (c) to the following set of sentences:

$$\begin{array}{l}A \Rightarrow B\\ B \Rightarrow C\\ C \Rightarrow \neg A\,.\end{array}$$

e. Since the algorithm in (b) is very similar to the algorithm for conversion to CNF, and since the algorithm in (c) is much simpler than any algorithm for solving a set of sentences in CNF, why is this technique not used in automated reasoning?

a. Each possible world can be expressed as the conjunction of all the literals that hold in the model. The sentence is then equivalent to the disjunction of all these conjunctions, i.e., a DNF expression.

b. A trivial conversion algorithm would enumerate all possible models and include terms corresponding to those in which the sentence is true; but this is necessarily exponential-time. We can convert to DNF using the same algorithm as for CNF except that we distribute $\wedge$ over $\vee$ at the end instead of the other way round.

c. A DNF expression is satisfiable if it contains at least one term that has no contradictory literals. This can be checked in linear time, or even during the conversion process. Any completion of that term, filling in missing literals, is a model.

d. The first steps give

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg A) .$$

Converting to DNF means taking one literal from each clause, in all possible ways, to generate the terms (8 in all). Choosing each literal corresponds to choosing the truth value of each variable, so the process is very like enumerating all possible models. Here, the first term is $(\neg A \wedge \neg B \wedge \neg C)$, which is clearly satisfiable.

e. The problem is that the final step typically results in DNF expressions of exponential size, so we require both exponential time AND exponential space.

Exercise 7.CCLX

Convert the following set of sentences to clausal form.

S1: $A \;\Leftrightarrow\; (B \vee E)$.
S2: $E \;\Rightarrow\; D$.
S3: $C \wedge F \;\Rightarrow\; \neg B$.
S4: $E \;\Rightarrow\; B$.
S5: $B \;\Rightarrow\; F$.
S6: $B \;\Rightarrow\; C$

Give a trace of the execution of DPLL on the conjunction of these clauses.

The CNF representations are as follows:

S1: $(\neg A \vee B \vee E) \wedge (\neg B \vee A) \wedge (\neg E \vee A)$.
S2: $(\neg E \vee D)$.
S3: $(\neg C \vee \neg F \vee \neg B)$.
S4: $(\neg E \vee B)$.
S5: $(\neg B \vee F)$.
S6: $(\neg B \vee C)$.

We omit the DPLL trace, which is easy to obtain from the version in the code repository.

Exercise 7.FCNF

Is a randomly generated 4-CNF sentence with n symbols and m clauses more or less likely to be solvable than a randomly generated 3-CNF sentence with n symbols and m clauses? Explain.

It is more likely to be solvable: adding literals to disjunctive clauses makes them easier to satisfy.

Exercise 7.MINE

Minesweeper, the well-known computer game, is closely related to the wumpus world. A minesweeper world is a rectangular grid of N squares with M invisible mines scattered among them. Any square may be probed by the agent; instant death follows if a mine is probed. Minesweeper indicates the presence of mines by revealing, in each probed square, the *number* of mines that are directly or diagonally adjacent. The goal is to probe every unmined square.

a. Let $X_{i,j}$ be true iff square $[i,j]$ contains a mine. Write down the assertion that exactly two mines are adjacent to [1,1] as a sentence involving some logical combination of $X_{i,j}$ propositions.

b. Generalize your assertion from (a) by explaining how to construct a CNF sentence asserting that k of n neighbors contain mines.

c. Explain precisely how an agent can use DPLL to prove that a given square does (or does not) contain a mine, ignoring the global constraint that there are exactly M mines in all.

d. Suppose that the global constraint is constructed from your method from part (b). How does the number of clauses depend on M and N? Suggest a way to modify DPLL so that the global constraint does not need to be represented explicitly.

e. Are any conclusions derived by the method in part (c) invalidated when the global constraint is taken into account?

f. Give examples of configurations of probe values that induce *long-range dependencies* such that the contents of a given unprobed square would give information about the contents of a far-distant square. (*Hint*: consider an $N \times 1$ board.)

a. This is a disjunction with 28 disjuncts, each one saying that two of the neighbors are true and the others are false. The first disjunct is

$$X_{2,2} \wedge X_{1,2} \wedge \neg X_{0,2} \wedge \neg X_{0,1} \wedge \neg X_{2,1} \wedge \neg X_{0,0} \wedge \neg X_{1,0} \wedge \neg X_{2,0}$$

The other 27 disjuncts each select two different $X_{i,j}$ to be true.

b. There will be $\binom{n}{k}$ disjuncts, each saying that k of the n symbols are true and the others false.

c. For each of the cells that have been probed, take the resulting number n revealed by the game and construct a sentence with $\binom{n}{8}$ disjuncts. Conjoin all the sentences together. Then use DPLL to answer the question of whether this sentence entails $X_{i,j}$ for the particular i, j pair you are interested in.

d. To encode the global constraint that there are M mines altogether, we can construct a disjunct with $\binom{M}{N}$ disjuncts, each of size N. Remember, $\binom{M}{N=M!/(M-N)!}$. So for a Minesweeper game with 100 cells and 20 mines, this will be morre than 10^{39}, and thus cannot be represented in any computer. However, we can represent the global constraint within the DPLL algorithm itself. We add the parameter *min* and *max* to the DPLL function; these indicate the minimum and maximum number of unassigned symbols that must be true in the model. For an unconstrained problem the values 0 and N will be used for these parameters. For a mineseeper problem the value M will be used for both *min* and *max*. Within DPLL, we fail (return false) immediately if *min* is less than the number of remaining symbols, or if *max* is less than 0. For each recursive call to DPLL, we update *min* and *max* by subtracting one when we assign a true value to a symbol.

e. No conclusions are invalidated by adding this capability to DPLL and encoding the global constraint using it.

f. Consider this string of alternating 1's and unprobed cells (indicated by a dash):

```
|-|1|-|1|-|1|-|1|-|1|-|1|-|1|-|
```

There are two possible models: either there are mines under every even-numbered dash, or under every odd-numbered dash. Making a probe at either end will determine whether cells at the far end are empty or contain mines.

Exercise 7.KNOW

How long does it take to prove $KB \models \alpha$ using DPLL when α is a literal *already contained in* KB? Explain.

It will take time proportional to the number of pure symbols plus the number of unit clauses. We assume that $KB \Rightarrow \alpha$ is false, and prove a contradiction. $\neg(KB \Rightarrow \alpha)$ is equivalent to $KB \land \neg\alpha$. From this sentence the algorithm will first eliminate all the pure symbols, then it will work on unit clauses until it chooses either α or $\neg\alpha$ (both of which are unit clauses); at that point it will immediately recognize that either choice (true or false) for α leads to failure, which means that the original non-negated assertion α is entailed.

Exercise 7.DPLL

Trace the behavior of DPLL on the knowledge base in Figure 7.16 when trying to prove Q, and compare this behavior with that of the forward-chaining algorithm.

We omit the DPLL trace, which is easy to obtain from the version in the code repository. The behavior is very similar: the unit-clause rule in DPLL ensures that all known atoms are propagated to other clauses.

Exercise 7.SSAL

Write a successor-state axiom for the *Locked* predicate, which applies to doors, assuming the only actions available are *Lock* and *Unlock*.

$$Locked^{t+1} \Leftrightarrow [Lock^t \lor (Locked^t \land \neg Unlock^t)]\ .$$

Exercise 7.OPTW

Discuss what is meant by *optimal* behavior in the wumpus world. Show that the HYBRID-WUMPUS-AGENT is not optimal, and suggest ways to improve it.

Optimal behavior for the Wumpus world is defined by the task environment in Section 7.2: optimal behavior maximizes the expected performance measure. We need to take an expectation because we are unsure of the initial state of the world.

The hybrid agent is not optimal as, for example, when it decides to risk a square with a pit it doesn't necessarily pick a square with minimal probability of having a pit: a breezy square that has three unvisited neighbours is safer to explore from than one which has two. One improvement, therefore, would explore from breezy squares with the most unexplored neighbours when there are no non-breezy square to explore from.

Exercise 7.TSWA

Suppose an agent inhabits a world with two states, S and $\neg S$, and can do exactly one of two actions, a and b. Action a does nothing and action b flips from one state to the other. Let S^t be the proposition that the agent is in state S at time t, and let a^t be the proposition that the agent does action a at time t (similarly for b^t).

a. Write a successor-state axiom for S^{t+1}.

b. Convert the sentence in (a) into CNF.

c. Show a resolution refutation proof that if the agent is in $\neg S$ at time t and does a, it will still be in $\neg S$ at time $t+1$.

a. $S^{t+1} \Leftrightarrow [(S^t \land a^t) \lor (\neg S^t \land b^t)]$.

b. Because the agent can do exactly one action, we know that $b^t \equiv \neg a^t$ so we replace b^t throughout. We obtain four clauses:
1: $(\neg S^{t+1} \lor S^t \lor \neg a^t)$
2: $(\neg S^{t+1} \lor \neg S^t \lor a^t)$
3: $(S^{t+1} \lor \neg S^t \lor \neg a^t)$
4: $(S^{t+1} \lor S^t \lor a^t)$

c. The goal is $(\neg S^t \wedge a^t) \Rightarrow \neg S^{t+1}$. Negated, this becomes three clauses: 5: $\neg S^t$; 6: a^t; 7: S^{t+1}. Resolving 5, 6, 7 against 1, we obtain the empty clause.

Exercise 7.SSAX

Section 7.7.1 provides some of the successor-state axioms required for the wumpus world. Write down axioms for all remaining fluent symbols.

The remaining fluents are the orientation fluents ($\mathit{FacingEast}$ etc.) and $\mathit{WumpusAlive}$. The successor-state axioms are as follows:

$$\begin{array}{rcl} \mathit{FacingEast}^{t+1} & \Leftrightarrow & (\mathit{FacingEast}^t \wedge \neg(\mathit{TurnLeft}^t \vee \mathit{TurnRight}^t)) \\ & \vee & (\mathit{FacingNorth}^t \wedge \mathit{TurnRight}^t) \\ & \vee & (\mathit{FacingSouth}^t \wedge \mathit{TurnLeft}^t) \\ \mathit{WumpusAlive}^{t+1} & \Leftrightarrow & \mathit{WumpusAlive}^t \wedge \neg(\mathit{WumpusAhead}^t \wedge \mathit{HaveArrow}^t \wedge \mathit{Shoot}^t) \,. \end{array}$$

The $\mathit{WumpusAhead}$ fluent does not need a successor-state axiom, since it is definable synchronously in terms of the agent location and orientation fluents and the wumpus location. The definition is extraordinarily tedious, illustrating the weakness of proposition logic. Note also that in the second edition we described a successor-state axiom (in the form of a circuit) for $\mathit{WumpusAlive}$ that used the Scream observation to infer the wumpus's death, with no need for describing the complicated physics of shooting. Such an axiom suffices for state estimation, but nor for planning.

Exercise 7.HYBR

Modify the HYBRID-WUMPUS-AGENT to use the 1-CNF logical state estimation method described on page 261. We noted on that page that such an agent will not be able to acquire, maintain, and use more complex beliefs such as the disjunction $P_{3,1} \vee P_{2,2}$. Suggest a method for overcoming this problem by defining additional proposition symbols, and try it out in the wumpus world. Does it improve the performance of the agent?

The required modifications are to add definitional axioms such as

$$P_{3,1\ or\ 2,2} \Leftrightarrow P_{3,1} \vee P_{2,2}$$

and to include the new literals on the list of literals whose truth values are to be inferred at each time step.

One natural way to extend the 1-CNF representation is to add test additional non-literal sentences. The sentences we choose to test can depend on inferences from the current KB. This can work if the number of additional sentences we need to test is not too large.

For example, we can query the knowledge base to find out which squares we know have pits, which we know might have pits, and which states are breezy (we need to do this to compute the un-augmented 1-CNF belief state). Then, for each breezy square, test the sentence "one of the neighbours of this square which might have a bit does have a pit." For example, this would test $P_{3,1} \vee P_{2,2}$ if we had perceived a breeze in square (2,1). Under the Wumpus physics, this literal will be true iff the breezy square has no known pit around it.