Propositional Logic:

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ De Morgan

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ De Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

PROPOSITIONAL THEOREM PROVING

. For example, the equivalence for biconditional elimination yields the two inference rules

There is no pit in [1,1]

$R_1 : \neg P_{1,1}$

A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R_4 : \neg B_{1,1}$.

$R_5 : B_{2,1}$

$R_6 : \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Then we apply And-Elimination to R_6 to obtain

$R_7 : \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Logical equivalence for contrapositives gives

$R_8 : \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$.

Now we can apply Modus Ponens with R_8 and the percept R_4 (i.e., $\neg B_{1,1}$), to obtain

$R_9 : \quad \neg(P_{1,2} \vee P_{2,1})$.

Finally, we apply De Morgan's rule, giving the conclusion

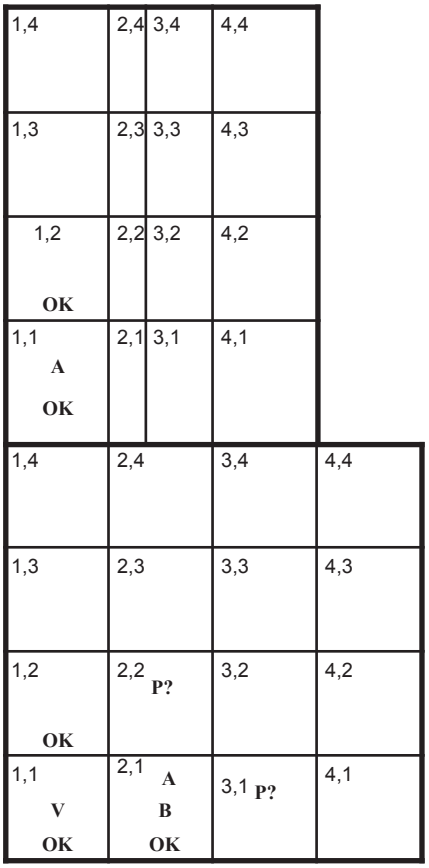
$R_{10} : \quad \neg P_{1,2} \wedge \neg P_{2,1}$.

One final property of logical systems is **monotonicity**, which says that the set of en- tailed sentences can only *increase* as information is added to the knowledge base. For any sentences α and β,

if $KB \models \alpha$ then $KB \wedge \beta \models \alpha$.

Proof by resolution

We begin by using a simple version of the resolution rule in the wumpus world. Let us consider the steps leading up to Figure 7.4(a): the agent returns from [2,1] to [1,1] and then goes to [1,2], where it perceives a stench, but no breeze. We add the following facts to the knowledge base:



R_{11}: $\neg B_{1,2}$.

R_{12}: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$.

By the same process that led to R_{10} earlier, we can now derive the absence of pits in [2,2] and [1,3] (remember that [1,1] is already known to be pitless):

R_{13}: $\neg P_{2,2}$.

R_{14}: $\neg P_{1,3}$.

We can also apply biconditional elimination to R_3, followed by Modus Ponens with R_5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1]:

R_{15}: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$.

Now comes the first application of the resolution rule: the literal $\neg P_{2,2}$ in R_{13} *resolves with* the literal $P_{2,2}$ in R_{15} to give the **resolvent**

R_{16}: $P_{1,1} \vee P_{3,1}$.

In English; if there's a pit in one of [1,1], [2,2], and [3,1] and it's not in [2,2], then it's in [1,1] or [3,1]. Similarly, the literal $\neg P_{1,1}$ in R_1 resolves with the literal $P_{1,1}$ in R_{16} to give

R_{17}: $P_{3,1}$.

In English: if there's a pit in [1,1] or [3,1] and it's not in [1,1], then it's in [3,1].

Conjunctive normal form

The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses. How, then, can it lead to a complete inference procedure for all of propositional logic? The answer is that *every sentence of propositional logic is logically equivalent to a conjunction of clauses.* A sentence expressed as a conjunction of clauses is said to be in **conjunctive normal form** or
CNF (see Figure 7.14). We now describe a procedure for converting to CNF. We illustrate the procedure by converting the sentence $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into CNF. The steps are as follows:

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$.
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

1. CNF requires $\neg$ to appear only in literals, so we "move $\neg$ inwards" by repeated appli- cation of the following equivalences from Figure 7.11:

   $\neg(\neg\alpha) \equiv \alpha$ (double-negation elimination)
   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (De Morgan)
   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (De Morgan)

   In the example, we require just one application of the last rule:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$.

2. Now we have a sentence containing nested $\wedge$ and $\vee$ operators applied to literals. We apply the distributivity law from Figure 7.11, distributing $\vee$ over $\wedge$ wherever possible.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$.

function PL-RESOLUTION(*KB*, α) **returns** *true* or *false*
inputs: *KB*, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of *KB* $\wedge \neg\alpha$
new ←{ }
loop do
  for each pair of clauses C_i, C_j **in** *clauses* **do**
    resolvents ← PL-RESOLVE(C_i, C_j)
    if *resolvents* contains the empty clause **then return** *true*
    new ← *new* ∪ *resolvents*
  if *new* ⊆ *clauses* **then return** *false*
  clauses ← *clauses* ∪ *new*

`PL-RESOLUTION(KB, α)` tries to decide whether the knowledge base (**KB**) *entails* the query (**α**).
Formally:

- We want to test if **KB ⊨ α**.
- This is equivalent to checking whether **KB ∧ ¬α** is *unsatisfiable*.

So the algorithm applies **resolution inference** until either:

1. It derives a contradiction (empty clause □) → **return true** (KB ⊨ α), or
2. No new information can be generated → **return false** (KB ⊭ α).

Horn clauses and definite clauses:

One such restricted form is the definite clause, which is a disjunction of literals of which exactly one is positive.
For example, the clause ($\neg L_{1,1} \vee \neg$Breeze $\vee B_{1,1}$) is a definite clause, whereas ($\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$) is not. Slightly more general is the Horn clause, which is a disjunction of literals of which at most one is positive. So all definite clauses are Horn clauses, as are clauses with no positive literals; these are called goal clauses.
Every definite clause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal. For example, the definite clause ($\neg L_{1,1} \vee \neg$Breeze $\vee B_{1,1}$) can be written as the implication ($L_{1,1} \wedge$ Breeze) $\Rightarrow B_{1,1}$. In the implication form, the sentence is easier to understand: it says that if the agent is in [1,1] and there is a breeze, then [1,1] is breezy. In Horn form, the premise is called the body and the conclusion is called the head.A sentence consisting of a single positive literal, such as $L_{1,1}$, is called a fact. It too can be written in implication form as True $\Rightarrow L_{1,1}$, but it is simpler to write just $L_{1,1}$

CNFSentence $\rightarrow$ Clause$_1 \wedge \cdots \wedge$Clause$_n$

Clause $\rightarrow$ Literal$_1 \vee \cdots \vee$Literal$_m$

Literal $\rightarrow$ Symbol | $\neg$Symbol

Symbol $\rightarrow$ P | Q| R| ...

HornClauseForm $\rightarrow$ DefiniteClauseForm | GoalClauseForm

DefiniteClauseForm $\rightarrow$ (Symbol$_1 \wedge \cdots \wedge$Symbol$_l$) $\Rightarrow$ Symbol

GoalClauseForm $\rightarrow$ (Symbol$_1 \wedge \cdots \wedge$Symbol$_l$) $\Rightarrow$ False

Agrammar for conjunctive normal form, Horn clauses, and definite clauses. A clause such as A $\wedge$ B $\Rightarrow$ C is still a definite clause when it is written as $\neg$A $\vee \neg$B $\vee$ C, but only the former is considered the canonical form for definite clauses. One more class is the k-CNF sentence, which is a CNF sentence where each clause has at most k literals.

/***The forward-chaining algorithm for propositional logic.*/**

function PL-FC-ENTAILS?(KB,q) returns true or false

inputs: KB, the knowledge base, a set of propositional definite clauses

q, the query, a proposition symbol

count ←a table,where count[c] is the number of symbols in c's premise

inferred ←a table,where inferred[s] is initially false for all symbols

agenda ←a queue of symbols, initially symbols known to be true in KB

while agenda is not empty do

p ←POP(agenda)

if p = q then return true

if inferred[p]=false then

inferred[p] ←true

for each clause c in KB where p is in c.PREMISE do

decrement count[c]

if count[c]=0then add c.CONCLUSION to agenda

return false