Foundations of Artificial Intelligence

7. Propositional Logic

Rational Thinking, Logic, Resolution

Joschka Boedecker and Wolfram Burgard and
Frank Hutter and Bernhard Nebel and Michael Tangermann

Albert-Ludwigs-Universität Freiburg

May 22, 2019

Motivation

Logic is a universal tool with many powerful applications

- Proving theorems
  - With the help of the algorithmic tools we describe here: automated theorem proving
- Formal verification
  - Verification of software
    - Ruling out unintended states (null-pointer exceptions, etc.)
    - Proving that the program computes the right solution
  - Verification of hardware (Pentium bug, etc)
- Basis for solving many NP-hard problems in practice
- Note: this and the next section (satisfiability) are based on Chapter 7 of the textbook ("Logical Agents")

Contents

Agents that Think Rationally

- Until now, the focus has been on agents that act rationally.
- Often, however, rational action requires rational (logical) thought on the agent's part.
- To that purpose, portions of the world must be represented in a knowledge base, or KB.
  - A KB is composed of sentences in a language with a truth theory (logic)
    - We (being external) can interpret sentences as statements about the world. (semantics)
    - Through their form, the sentences themselves have a causal influence on the agent's behavior. (syntax)
- Interaction with the KB through $\textsc{Ask}$ and $\textsc{Tell}$ (simplified):

$\textsc{Ask}(\text{KB},\alpha) = \text{yes}$	exactly when α follows from the KB
$\textsc{Tell}(\text{KB},\alpha) = \text{KB'}$	so that α follows from KB'
$\textsc{Forget}(\text{KB},\alpha) = \text{KB'}$	non-monotonic (will not be discussed)

3 Levels

In the context of knowledge representation, we can distinguish three levels [Newell 1990]:

Knowledge level: Most abstract level. Concerns the total knowledge contained in the KB. For example, the automated DB information system knows that a trip from Freiburg to Basel SBB with an ICE costs 24.70 €.

Logical level: Encoding of knowledge in a formal language.
$Price(Freiburg, Basel, 24.70)$

Implementation level: The internal representation of the sentences, for example:

- As a string ``Price(Freiburg, Basel, 24.70)''
- As a value in a matrix

When ASK and TELL are working correctly, it is possible to remain on the knowledge level. Advantage: very comfortable user interface. The user has his/her own mental model of the world (statements about the world) and communicates it to the agent (TELL).

A Knowledge-Based Agent

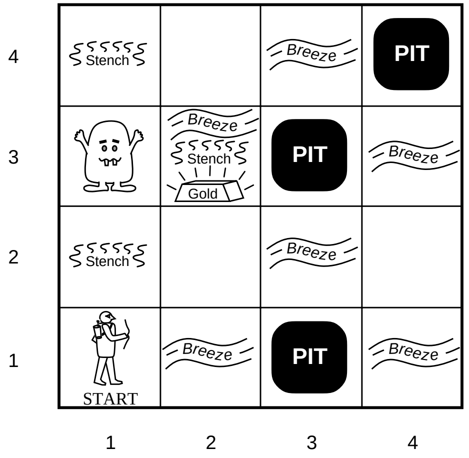
A knowledge-based agent uses its knowledge base to

- represent its background knowledge
- store its observations
- store its executed actions
- . . . derive actions

```
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
```

The Wumpus World (1): Illustration



This is just one sample configuration.

The Wumpus World (2)

- A 4×4 grid
- In the square containing the wumpus and in the directly adjacent squares, the agent perceives a stench.
- In the squares adjacent to a pit, the agent perceives a breeze.
- In the square where the gold is, the agent perceives a glitter.
- When the agent walks into a wall, it perceives a bump.
- When the wumpus is killed, its scream is heard everywhere.
- Percepts are represented as a 5-tuple, e.g.,

$$[Stench, Breeze, Glitter, None, None]$$

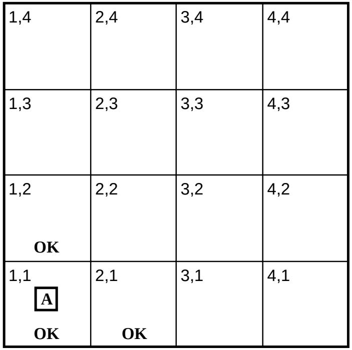
means that it stinks, there is a breeze and a glitter, but no bump and no scream. The agent *cannot* perceive its own location, cannot look in adjacent square.

The Wumpus World (3)

- Actions: Go forward, turn right by 90°, turn left by 90°, pick up an object in the same square (grab), shoot (there is only one arrow), leave the cave (only works in square [1,1]).
- The agent dies if it falls down a pit or meets a live wumpus.
- Initial situation: The agent is in square [1,1] facing east. Somewhere exists a wumpus, a pile of gold and 3 pits.
- Goal: Find the gold and leave the cave.

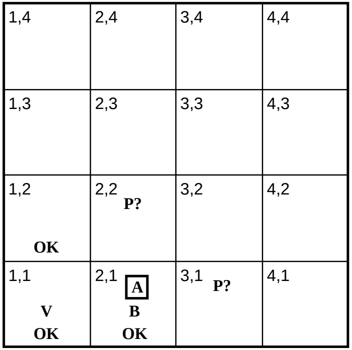
The Wumpus World (4)

[1,2] and [2,1] are safe:



(a)

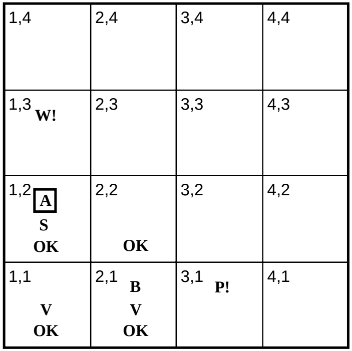
$\boxed{A}$	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*



(b)

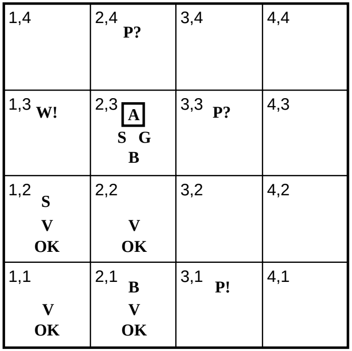
The Wumpus World (5)

The wumpus is in [1,3]!



(a)

$\boxed{\text{A}}$	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*



(b)

Syntax and Semantics

- Knowledge bases consist of sentences
- Sentences are expressed according to the syntax of the representation language
  - Syntax specifies all the sentences that are well-formed
  - E.g., in ordinary arithmetic, syntax is pretty clear:

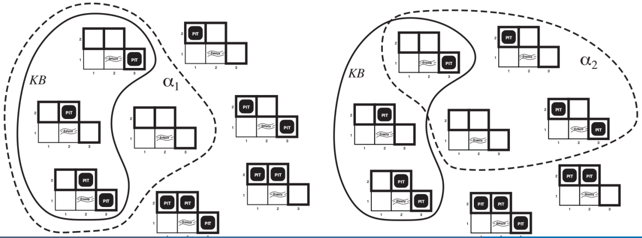
    $x + y = 4$ is a well-formed sentence
    $x4y+ =$ is not a well-formed sentence
- A logic also defines the semantics or meaning of sentences
  - Defines the truth of a sentence with respect to each possible world
  - E.g., specifies that the sentence $x + y = 4$ is true in a world in which $x = 2$ and $y = 2$, but not in a world in which $x = 1$ and $y = 1$

Logical Entailment

- If a sentence α is true in a possible world m, we say that m satisfies α or m is a model of α
- We denote the set of all models of α by $M(\alpha)$
- Logical entailment:
  - When does a sentence β logically follow from another sentence α?
    - in symbols $\alpha \models \beta$
  - $\alpha \models \beta$ if and only if (iff) in every model in which α is true, β is also true
    - I.e., $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$
    - α is a *stronger* assertion than β; it rules out *more* possible worlds
  - Example in arithmetic: sentence $x = 0$ entails sentence $xy = 0$
    - $x = 0$ rules out the possible world $\{x = 1, y = 0\}$, whereas $xy = 0$ does not rule out that world

Example in the Wumpus World

- Which worlds are possible after having visited [1,1] (no breeze) and [2,1] (breeze)?
  - all worlds in solid area
- Consider two possible sentences:
  - $\alpha_1 =:$ "There is no pit in [1,2]" (true in models in dashed area below, left)
  - $\alpha_2 =:$ "There is no pit in [2,2]" (true in models in dashed area below, right)
- KB $\models \alpha_1$
  - By inspection: in every model in which KB is true, α_1 is also true
- KB $\not\models \alpha_2$
  - In some models, in which KB is true, α_2 is false



Entailment and Inference

- Logical entailment is the (semantic) relation between models of the KB (or a set of formulae in general) and models of a sentence.
- How can we procedurally generate/derive entailed sentences?
  - Logical entailment: $KB \models \alpha$
  - Inference: we can derive α with an inference method i.
    This is written as: $KB \vdash_i \alpha$
- We'd like to have inference algorithms that derive only sentences that are entailed (soundness) and all of them (completeness)

Declarative Languages

Before a system that is capable of learning, thinking, planning, explaining, . . . can be built, one must find a way to express knowledge.

We need a precise, declarative language.

- Declarative
  - We state what we want to compute, not how
  - System believes P if and only if (iff) it considers P to be true
- Precise: We must know,
  - which symbols represent sentences,
  - what it means for a sentence to be true, and
  - when a sentence follows from other sentences.

One possibility: Propositional Logic

Basics of Propositional Logic (1)

Propositions: The building blocks of propositional logic are indivisible, atomic statements (atomic propositions), e.g.,

- "The block is red", expressed, e.g., by the symbol "B_{red}"
- "The wumpus is in [1,3]", expressed, e.g., by the symbol "$W_{1,3}$"

and the logical connectives "and", "or", and "not", which we can use to build formulae.

Basics of Propositional Logic (2)

We are interested in knowing the following:

- When is a proposition true?
- When does a proposition follow from a knowledge base (KB)?
  - Symbolically: KB $\models \varphi$
- Can we (syntactically) define the concept of *derivation*,
  - Symbolically: KB $\vdash \varphi$
- And can we make sure that $\models$ and $\vdash$ are equivalent?

$\rightarrow$ Meaning and implementation of $\textsc{Ask}$

Syntax of Propositional Logic

Countable alphabet Σ of atomic propositions: P, Q, R, $W_{1,3}$, ...

Logical formulae:		
	$P \in \Sigma$	atomic formula
	$\bot$	falseness
	$\top$	truth
	$\neg\varphi$	negation
	$\varphi \land \psi$	conjunction
	$\varphi \lor \psi$	disjunction
	$\varphi \Rightarrow \psi$	implication
	$\varphi \Leftrightarrow \psi$	equivalence

Operator precedence: $\neg > \land > \lor > \Rightarrow > \Leftrightarrow$. (use brackets when necessary)

Atom: atomic formula
Literal: (possibly negated) atomic formula
Clause: disjunction of literals

Semantics: Intuition

Atomic propositions can be true (T) or false (F).

The truth of a formula follows from the truth of its atomic propositions (truth assignment or interpretation) and the connectives.

Example:

$$(P \vee Q) \wedge R$$

- If P and Q are *false* and R is *true*, the formula is *false*
- If P and R are *true*, the formula is *true* regardless of what Q is.

Semantics: Formally

A truth assignment of the atoms in Σ, or an interpretation I over Σ, is a function

$$I : \Sigma \mapsto \{T, F\}$$

Interpretation I satisfies a formula φ ('$I \models \varphi$'):

$$\begin{array}{lll}
I \models \top & & \\
I \not\models \bot & & \\
I \models P & \text{iff} & P^I = T \\
I \not\models \neg\varphi & \text{iff} & I \models \varphi \\
I \models \varphi \wedge \psi & \text{iff} & I \models \varphi \text{ and } I \models \psi \\
I \models \varphi \vee \psi & \text{iff} & I \models \varphi \text{ or } I \models \psi \\
I \models \varphi \Rightarrow \psi & \text{iff} & \text{if } I \models \varphi\text{, then } I \models \psi \\
I \models \varphi \Leftrightarrow \psi & \text{iff} & \text{if } I \models \varphi \text{ if and only if } I \models \psi
\end{array}$$

I satisfies φ ($I \models \varphi$) or φ is true under I, when $I(\varphi) = T$.
I can be seen as a 'possible world'

Example

$$I : \begin{cases} P \mapsto T \\ Q \mapsto T \\ R \mapsto F \\ S \mapsto F \\ \dots \end{cases}$$

$$\varphi = ((P \vee Q) \Leftrightarrow (R \vee S)) \wedge (\neg(P \wedge Q) \wedge (R \wedge \neg S))$$

Question: $I \models \varphi$?

Wumpus World in Propositional Logic

Symbols: $B_{1,1}, B_{1,2}, \ldots, B_{2,1}, \ldots, S_{1,1}, \ldots, P_{1,1}, \ldots, W_{1,1}, \ \ldots$
Meaning: $B = \mathit{Breeze}$, $B_{i,j} =$ there is a breeze in (i, j) etc.

Facts and Rules:

R1: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R2: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R3: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
...
F1: $\neg P_{1,1}$
F2: $\neg B_{1,1}$ (no percept in (1,1))
F3: $B_{2,1}$ (percept)
F4: $\neg B_{1,2}$ (no percept)
...

Terminology

An interpretation I is called a model of φ if $I \models \varphi$.

An interpretation is a model of a set of formulae if it satisfies all formulae of the set.

A formula φ is

- satisfiable if there exists I that satisfies φ,
- unsatisfiable if φ is not satisfiable,
- falsifiable if there exists I that doesn't satisfy φ, and
- valid (a tautology) if $I \models \varphi$ holds for all I.

Question for you: how are these related to each other?

Two formulae are

- logically equivalent ($\varphi \equiv \psi$) if $I \models \varphi$ iff $I \models \psi$ holds for all I.

The Truth Table Method

How can we decide if a formula is satisfiable, valid, etc.?

$\rightarrow$ Generate a truth table

Example: Is $\varphi = ((P \vee H) \wedge \neg H) \Rightarrow P$ valid?

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
F	F	F	F	T
F	T	T	F	T
T	F	T	T	T
T	T	T	F	T

Since the formula is true for all possible combinations of truth values (satisfied under all interpretations), φ is valid.

Satisfiability, falsifiability, unsatisfiability likewise.

Logical Implications

Goal: Find an algorithmic way to derive new knowledge out of a knowledge base

1. Transform KB into a standardized representation
2. define rules that syntactically modify formulae while keeping semantic correctness

Normal Forms

- A formula is in conjunctive normal form (CNF) if it consists of a conjunction of disjunctions of literals $l_{i,j}$, i.e., if it has the following form:

$$\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} l_{i,j} \right)$$

- A formula is in disjunctive normal form (DNF) if it consists of a disjunction of conjunctions of literals:

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$$

- For every formula, there exists at least one equivalent formula in CNF and one in DNF.
- A formula in DNF is satisfiable iff one disjunct is satisfiable.
  - Checking satisfiability of DNF formula takes linear time.
- A formula in CNF is valid iff every conjunct is valid.
  - Checking validity of CNF formula takes linear time.

Producing CNF

1. Eliminate $\Rightarrow$ and $\Leftrightarrow$: $\alpha \Rightarrow \beta \rightarrow (\neg\alpha \lor \beta)$ etc.
2. Move $\neg$ inwards: $\neg(\alpha \land \beta) \rightarrow (\neg\alpha \lor \neg\beta)$ etc. (De Morgan's laws)
3. Distribute $\lor$ over $\land$: $((\alpha \land \beta) \lor \gamma) \rightarrow (\alpha \lor \gamma) \land (\beta \lor \gamma)$
4. Simplify: $\alpha \lor \alpha \rightarrow \alpha$ etc.

The result is a conjunction of disjunctions of literals (CNF)

An analogous process converts any formula to an equivalent formula in DNF.

- During conversion, formulae can expand *exponentially*.
- Note: Conversion to CNF formula can be done *polynomially* if only satisfiability should be preserved

Logical Implication: Intuition

A set of formulae (a KB) usually provides an incomplete description of the world, i.e., it leaves the truth values of certain propositions open.

Example: KB $= \{(P \vee Q) \wedge (R \vee \neg P) \wedge S\}$ is definitive with respect to S, but leaves P, Q, R open (although they cannot take on arbitrary values).

Models of the KB:

P	Q	R	S
F	T	F	T
F	T	T	T
T	F	T	T
T	T	T	T

In all models of the KB, $Q \vee R$ is true, i.e., $Q \vee R$ follows logically from KB.

Logical Implication: Formal

The formula φ follows logically from a KB if φ is true in all models of the KB (symbolically KB $\models \varphi$):

$$\text{KB} \models \varphi \text{ iff } I \models \varphi \text{ for all models } I \text{ of KB}$$

Note: The $\models$ symbol is a *meta-symbol*

Question: Can we determine KB $\models \varphi$ without considering all interpretations (the truth table method)?

Some properties of logical implication relationships:

- Deduction theorem: $\text{KB} \cup \{\varphi\} \models \psi$ iff $\text{KB} \models \varphi \Rightarrow \psi$
- Contraposition theorem: $\text{KB} \cup \{\varphi\} \models \neg\psi$ iff $\text{KB} \cup \{\psi\} \models \neg\varphi$
- Contradiction theorem: $\text{KB} \cup \{\varphi\}$ is unsatisfiable iff $\text{KB} \models \neg\varphi$

Proof of the Deduction Theorem

Deduction theorem: $\mathsf{KB} \cup \{\varphi\} \models \psi$ iff $\mathsf{KB} \models \varphi \Rightarrow \psi$

"$\Rightarrow$" Assumption: $\mathsf{KB} \cup \{\varphi\} \models \psi$, i.e., every model of $\mathsf{KB} \cup \{\varphi\}$ is also a model of ψ.

Let I be any model of KB. If I is also a model of φ, then it follows that I is also a model of ψ.

This means that I is also a model of $\varphi \Rightarrow \psi$, i.e., $\mathsf{KB} \models \varphi \Rightarrow \psi$.

"$\Leftarrow$" Assumption: $\mathsf{KB} \models \varphi \Rightarrow \psi$. Let I be any model of KB that is also a model of φ, i.e., $I \models \mathsf{KB} \cup \{\varphi\}$.

From the assumption, I is also a model of $\varphi \Rightarrow \psi$ and thereby also of ψ , i.e., $\mathsf{KB} \cup \{\varphi\} \models \psi$.

Proof of the Contraposition Theorem

Contraposition theorem: KB $\cup \{\varphi\} \models \neg\psi$ iff KB $\cup \{\psi\} \models \neg\varphi$

$$
\begin{aligned}
& \text{KB} \cup \{\varphi\} \models \neg\psi \\
& \text{iff KB} \models \varphi \Rightarrow \neg\psi && (1) \\
& \text{iff KB} \models (\neg\varphi \vee \neg\psi) \\
& \text{iff KB} \models (\neg\psi \vee \neg\varphi) \\
& \text{iff KB} \models \psi \Rightarrow \neg\varphi \\
& \text{iff KB} \cup \{\psi\} \models \neg\varphi && (2)
\end{aligned}
$$

Note:
(1) and (2) are applications of the deduction theorem.

Inference Rules, Calculi, and Proofs

We can often derive new formulae from formulae in the KB. These new formulae should follow logically from the syntactical structure of the KB formulae.

Example: If KB $= \{\ldots, (\varphi \Rightarrow \psi), \ldots, \varphi, \ldots\}$ then ψ is a logical consequence of KB.

$\rightarrow$ Inference rules, e.g., $\dfrac{\varphi, \varphi \Rightarrow \psi}{\psi}$.

Calculus: Set of inference rules (potentially including so-called logical axioms).

Proof step: Application of an inference rule on a set of formulae.

Proof: Sequence of proof steps where every newly-derived formula is added, and in the last step, the goal formula is produced.

Soundness and Completeness

In the case where in the calculus C there is a proof for a formula φ, we write

$$\mathsf{KB} \vdash_C \varphi$$

(optionally without subscript C).

A calculus C is sound (or correct) if all formulae that are derivable from a KB actually follow logically.

$$\mathsf{KB} \vdash_C \varphi \text{ implies } \mathsf{KB} \models \varphi$$

This normally follows from the soundness of the inference rules and the logical axioms.

A calculus is complete if every formula that follows logically from the KB is also derivable with C from the KB:

$$\mathsf{KB} \models \varphi \text{ implies } \mathsf{KB} \vdash_C \varphi$$

Resolution: Idea

We want a way to derive new formulae that does not depend on testing every interpretation.

Idea: To prove that $\text{KB} \models \varphi$, we can prove that $\text{KB} \cup \{\neg\varphi\}$ is unsatisfiable (contradiction theorem). Therefore, in the following we attempt to show that a set of formulae is unsatisfiable.

Condition: All formulae must be in CNF.

However: In most cases, the formulae are close to CNF (and there exists a fast satisfiability-preserving transformation - Theoretical Computer Science course).

Nevertheless: In the worst case, this derivation process requires an exponential amount of time (this is, however, probably unavoidable).

Resolution: Representation

Assumption: All formulae in the KB are in CNF.

Equivalently, we can assume that the KB is a *set of clauses*. E.g.: Replace $\{(P \vee Q) \wedge (R \vee \neg P) \wedge S\}$ by $\{\{P, Q\}, \{R, \neg P\}, \{S\}\}$

Due to commutativity, associativity, and idempotence of $\vee$, clauses can also be understood as sets of literals. The empty set of literals is denoted by $\square$.

Set of clauses: Δ

Set of literals: C, D

Literal: l

Negation of a literal: $\bar{l}$

An interpretation I satisfies C iff there exists $l \in C$ such that $I \models l$. I satisfies Δ if for all $C \in \Delta : I \models C$, i.e., $I \not\models \square$, $I \not\models \{\square\}$, for all I.

The Resolution Rule

$$\frac{C_1 \dot\cup \{l\}, C_2 \dot\cup \{\bar{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ are called resolvents of the parent clauses $C_1 \dot\cup \{l\}$ and $C_2 \dot\cup \{\bar{l}\}$. l and $\bar{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not equivalent to the parent clauses, but it follows from them!

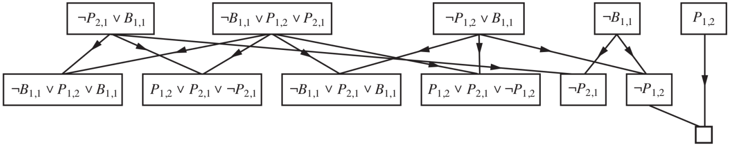
Notation: $R(\Delta) = \Delta \cup \{C \mid C$ is a resolvent of two clauses from $\Delta\}$

The Resolution Algorithm

```
function PL-RESOLUTION($KB, \alpha$) returns $true$ or $false$
  inputs: $KB$, the knowledge base, a sentence in propositional logic
          $\alpha$, the query, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \land \neg\alpha$
  $new \leftarrow \{\}$
  loop do
     for each pair of clauses $C_i, C_j$ in $clauses$ do
        $resolvents \leftarrow$ PL-RESOLVE($C_i, C_j$)
        if $resolvents$ contains the empty clause then return $true$
        $new \leftarrow new \cup resolvents$
     if $new \subseteq clauses$ then return $false$
     $clauses \leftarrow clauses \cup new$
```

Example:


$\neg P_{2,1} \lor B_{1,1}$
$\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$
$\neg P_{1,2} \lor B_{1,1}$
$\neg B_{1,1}$
$P_{1,2}$
$\neg B_{1,1} \lor P_{1,2} \lor B_{1,1}$
$P_{1,2} \lor P_{2,1} \lor \neg P_{2,1}$
$\neg B_{1,1} \lor P_{2,1} \lor B_{1,1}$
$P_{1,2} \lor P_{2,1} \lor \neg P_{1,2}$
$\neg P_{2,1}$
$\neg P_{1,2}$

Resolution: Soundness

We say D can be derived from Δ using resolution, i.e.,

$$\Delta \vdash D,$$

if there exist C_1, C_2, C_3, ..., $C_n = D$ such that

$$C_i \in R(\Delta \cup \{C_1, \dots, C_{i-1}\}), \text{ for } 1 \leq i \leq n.$$

Lemma (soundness) If $\Delta \vdash D$, *then* $\Delta \models D$.

Proof idea: Since all $D \in R(\Delta)$ *follow logically from* Δ, the Lemma results through induction over the length of the derivation.

Resolution: Completeness?

Is resolution also complete, i.e., is

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi$$

valid? Not in general. For example, consider:

$$\{\{a, b\}, \{\neg b, c\}\} \models \{a, b, c\} \nvdash \{a, b, c\}$$

However, it can be shown that resolution is refutation-complete: Δ is unsatisfiable implies $\Delta \vdash \square$

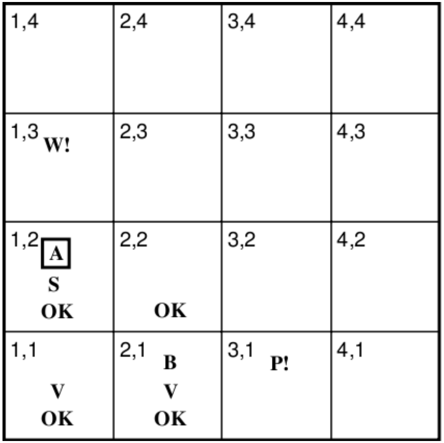
Theorem: Δ is unsatisfiable iff $\Delta \vdash \square$

With the help of the contradiction theorem, we can show that KB $\models \varphi$.
Idea: KB $\cup \{\neg\varphi\}$ is unsatisfiable iff KB $\models \varphi$

Resolution: Overview

- Resolution is a refutation-complete proof process. There are others (Davis-Putnam Procedure, Tableaux Procedure, . . .).
- In order to implement the process, a strategy must be developed to determine which resolution steps will be executed and when.
- In the worst case, a resolution proof can take exponential time. This, however, very probably holds for all other proof procedures.
- For CNF formulae in propositional logic, the fastest complete algorithms are indeed based on resolution (combined with backtracking search)

Where is the Wumpus? The Situation



$\boxed{A}$	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*

Where is the Wumpus? Knowledge of the Situation

$B = Breeze$, $S = Stench$, $B_{i,j} =$ there is a breeze in (i, j)

$\neg S_{1,1} \quad \neg B_{1,1}$
$\neg S_{2,1} \quad B_{2,1}$
$S_{1,2} \quad \neg B_{1,2}$

Knowledge about the wumpus and smell:

$R_1 : \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
$R_2 : \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$
$R_3 : \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$
$R_4 : S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

To show: KB $\models W_{1,3}$

Clausal Representation of the Wumpus World

Situational knowledge:
$\neg S_{1,1}$, $\neg S_{2,1}$, $S_{1,2}$

Knowledge of rules:
Knowledge about the wumpus and smell:
$R_1 : S_{1,1} \vee \neg W_{1,1},\ S_{1,1} \vee \neg W_{1,2},\ S_{1,1} \vee \neg W_{2,1}$
$R_2 : \ldots,\ S_{2,1} \vee \neg W_{2,2},\ \ldots$
$R_3 : \ldots$
$R_4 : \neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
$\ldots$

Negated goal formula: $\neg W_{1,3}$

Resolution Proof for the Wumpus World

Resolution:

$\neg W_{1,3},\ \neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
$\rightarrow \neg S_{1,2} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

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

$\neg S_{1,1},\ S_{1,1} \vee \neg W_{1,1}$
$\rightarrow \neg W_{1,1}$

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

...

$\neg W_{2,2},\ W_{2,2}$
$\rightarrow \square$

From Knowledge to Action

We can now infer new facts, but how do we translate knowledge into action?

Negative selection: Excludes any provably dangerous actions.

$$A_{1,1} \wedge East_A \wedge W_{2,1} \Rightarrow \neg Forward$$

Positive selection: Only suggests actions that are provably safe.

$$A_{1,1} \wedge East_A \wedge \neg W_{2,1} \Rightarrow Forward$$

Differences?

From the suggestions, we must still select an action.

Problems with Propositional Logic

Although propositional logic suffices to represent the wumpus world, it is rather involved.

Rules must be set up for each square.
$R_1 : \neg S_{1,1} \Rightarrow \neg W_{1,1} \land \neg W_{1,2} \land \neg W_{2,1}$
$R_2 : \neg S_{2,1} \Rightarrow \neg W_{1,1} \land \neg W_{2,1} \land \neg W_{2,2} \land \neg W_{3,1}$
$R_3 : \neg S_{1,2} \Rightarrow \neg W_{1,1} \land \neg W_{1,2} \land \neg W_{2,2} \land \neg W_{1,3}$
...

We need a time index for each proposition to represent the validity of the proposition over time $\rightarrow$ further expansion of the rules.

$\rightarrow$ More powerful logics exist, in which we can use object variables.
$\rightarrow$ First-Order Predicate Logic

Summary

- Rational agents require knowledge of their world in order to make rational decisions.
- With the help of a declarative (knowledge-representation) language, this knowledge is represented and stored in a knowledge base.
- We use propositional logic for this (for the time being).
- Formulae of propositional logic can be valid, satisfiable, or unsatisfiable.
- The concept of logical implication is important.
- Logical implication can be mechanized by using an inference calculus $\rightarrow$ resolution.
- Propositional logic quickly becomes impractical when the world becomes too large (or infinite).