UNIVERSITÄT
DES
SAARLANDES

Artificial Intelligence

Propositional Logic

Prof. Dr. habil. Jana Koehler

Dr. Sophia Saller, M. Sc. Annika Engel

Summer 2020

Deep thanks goes to
Prof. Jörg Hoffmann for
sharing his course material

Recommended Reading

- AIMA Chapter 7: Logical Agents

- Read more in: Handbook of Satisfiability
  - Edited by Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh, IOS Press 2009
  - Chapters 1-4
  - https://epdf.pub/queue/handbook-of-satisfiability.html

Agenda

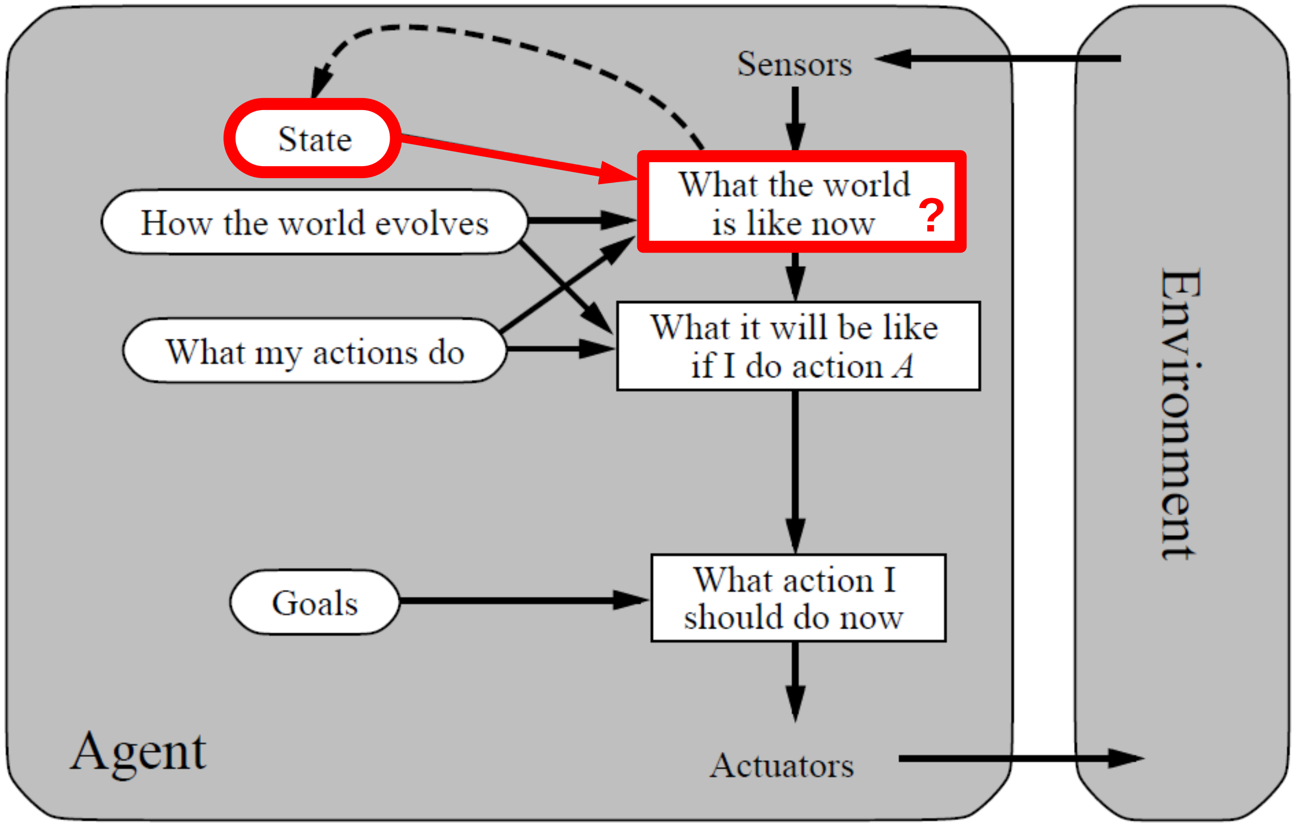
- Propositional Logic
  - Syntax
  - Semantics
  - Conjunctive and Disjunctive Normal Forms

- Reasoning in Propositional Logic
  - Basic Terminology
  - Resolution
  - Davis-Putnam Logemann-Loveland Algorithm
  - Conflict-Driven Clause Learning

Reasoning Capabilites in a Goal-based Agent

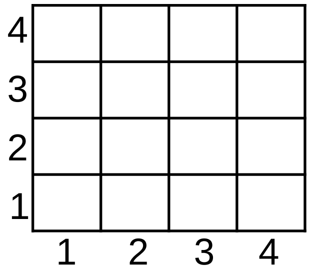
(1) Ability to perceive environment

(2) Observations used to make Decisions


Sensors
State
What the world is like now
?
How the world evolves
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Towards An Agent´s World Model - The Wumpus World

- A grid world with gold and pits



- Habitated by the **Wumpus**
  - Dangerous! Guards the gold and can attack the agent when trying to fetch the gold (world is underspecified in textbook)
  - Noticable by the agent through stench in adjacent cells

- ***Agent***
  - Armed with 1 arrow
  - With the goal to find the gold and leave the cave alive

Rules of the Wumpus World

- Agent dies if it steps onto cell with Wumpus or cell with pit
- Agent can only shoot into direction it is facing
- Cave can only be left via cell (1,1)
- Wumpus is in a cell next to gold to guard it (we cannot enter the field with the gold if the Wumpus is alive)
- Locations of pits, gold, and Wumpus are initially unknown to the agent

Percepts of the Agent

- Stench
  - Only when on a cell adjacent to the Wumpus
- Breeze
  - Only when on a cell adjacent to a pit
- Glitter
  - Only when on the cell with the gold
- Bump
  - Only when walking into a wall
- Scream
  - Only when Wumpus hit by arrow

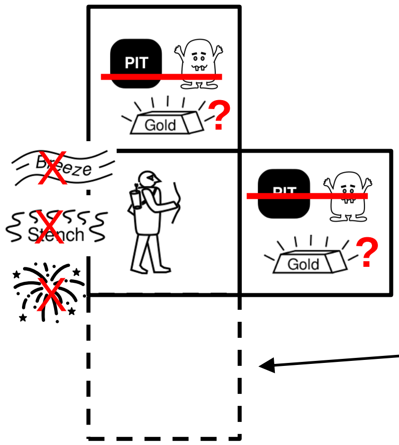
Actions of the Agent

- Go forward one cell in direction facing
- Turn right (by 90 degrees)
- Turn left (by 90 degrees)
- Grab object in current cell
- Shoot arrow

How can the agent safely explore the cave and find the gold?

Initial State: What the Agent Knows Initially

- Agent is in cell next to the entry of the cave, facing east
- Percepts: no stench, no breeze, no glitter
- Conclusions of the agent: neighboring cells are safe
  - No Wumpus, no pits, no gold



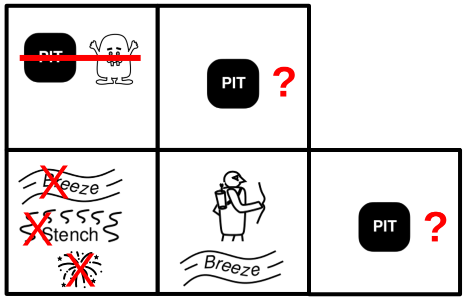
How can the agent know that there is no cell in this direction?

- ➢ TurnRight, MoveForward
- ➢ „bump“ ☹

We omit the bump actions from the discussion…

Move Forward

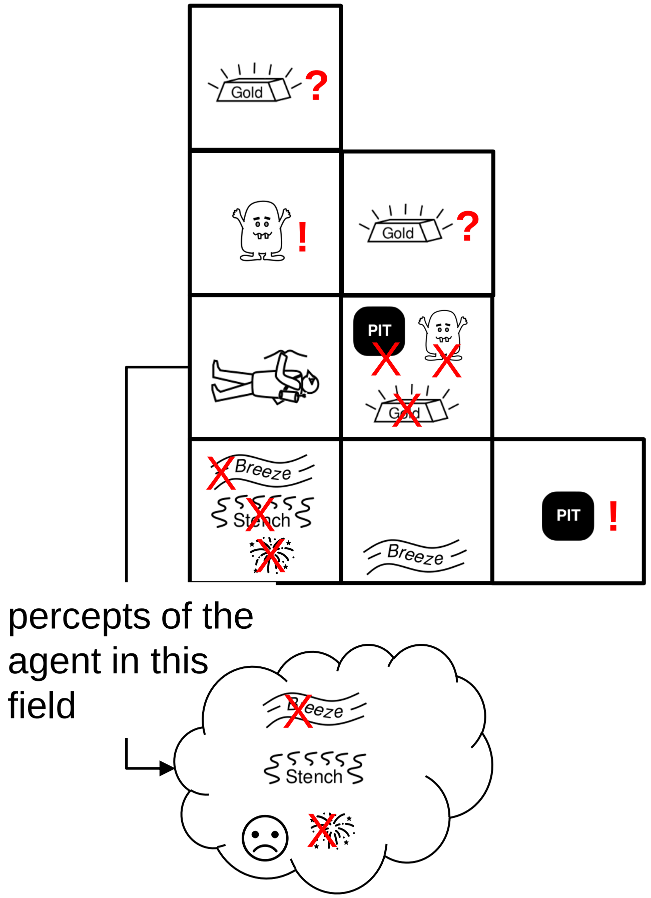
- Moving forward is one of the 2 safe actions
  - let us assume the agent moves to cell (2,1)
- The agent agent notices a breeze
  - a pit must be in a neighboring cell
- The agent does not notice glitter or stench
  - no Wumpus in a neighboring cell, no gold in this cell



➢ Agent cannot move safely to the next neighboring cells
➢ Need to back up and explore remaining safe cell

TurnLeft - TurnLeft - Move Forward - TurnRight - MoveForward

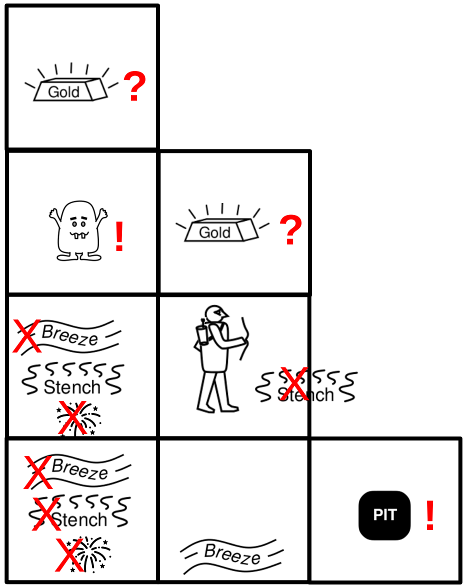
- Agent notices no breeze, no glitter, but stench
- The Wumpus is in the direction the agent is facing
  - Because the agent did not notice stench earlier
  - There is no cell to the left of the agent (remember we omit the bumps from the discussion ☹)
- There is no pit in the cell right to the agent
  - Because we do not notice a breeze

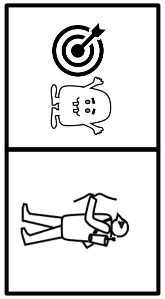


TurnRight - MoveForward

- The agent notices no stench
  - The Wumpus must be in the upper cell
- TurnLeft -TurnLeft - MoveForward - TurnRight - Shoot Arrow
  - ...

- Now the agent can move to the cell with the gold, grab the gold and leave the cave





Rationally Thinking Agent

```
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))
    $t \leftarrow t + 1$
    return action
```

- Mathematical Logic provides us with a method to formalize the „thinking“ of the agent
  - Keep a knowledge base represented in logic
  - Automate thinking by a logical calculus

The Calculus of Propositional Logic

- Syntax
  - What are structurally legal statements (formulas) in this logic?
- Semantics
  - What is the meaning of formulas?
- Calculus
  - Which rules and methods allow us to reason about formulas?
    - Find out if a formula/set of formulas is true or false?
    - Find out if a formula follows from another formula or set of formulas?

Syntax

Let Σ be a set of boolean variables (atomic propositions). Then

1. **F** ($\bot$) and **T** ($\top$) are Σ-formulas („*false*“, „*true*“).
2. Each $P \in \Sigma$ is a Σ-formula.
3. If φ is a Σ-formula, then so is $\neg\varphi$ („negation“).
4. If φ and ψ are Σ-formulas, then so are
   a) $\varphi \wedge \psi$ („conjunction“)
   b) $\varphi \vee \psi$ („disjunction“)
   c) $\varphi \Rightarrow \psi$ („implication“)
   d) $\varphi \Leftrightarrow \psi$ („equivalence“).

Atoms and negated atoms are called ***literals***.

BNF Grammar for Propositional Logic

$$
\begin{array}{ll}
Sentence & \rightarrow AtomicSentence \mid ComplexSentence \\
AtomicSentence & \rightarrow True \mid False \mid P \mid Q \mid R \mid \ldots \\
ComplexSentence & \rightarrow (Sentence) \mid [Sentence] \\
 & \mid \neg Sentence \\
 & \mid Sentence \wedge Sentence \\
 & \mid Sentence \vee Sentence \\
 & \mid Sentence \Rightarrow Sentence \\
 & \mid Sentence \Leftrightarrow Sentence
\end{array}
$$

Operator Precedence: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

implication / rule / if-then-else statement

$$\begin{tikzcd}
\text{antecedent / premise} \arrow[r] & \varphi \Rightarrow \psi & \text{consequent / conclusion} \arrow[l]
\end{tikzcd}$$

Literal, Clause, and Sentence

- **Literal**
  - atom
  - negation of an atom

  Examples: x_1, $\neg x_1$

- **Clause**
  - disjunction of literals

  Examples: $(x_1 \lor x_2)$, $(\neg x_1 \lor x_2 \lor \neg x_3)$

- **Sentence**
  - any atomic (literal) or complex proposition/sentence

  Examples: $(x_1 \lor x_2) \Rightarrow (\neg x_1 \lor x_2 \lor \neg x_3)$, $(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_3)$

Semantics

- Defines the rules for determining the truth of a proposition/ sentence with respect to an interpretation
- The interpretation assigns truth values to every proposition

Let Σ be a set of propositions. An ***interpretation*** I of Σ, also called a ***truth assignment***, is a function

$$I : \Sigma \rightarrow \{0, 1\}$$

Definition of I

$I \vDash \top$		
$I \nvDash \bot$		
$I \vDash P$	iff	$P^I = \text{true}$
$I \vDash \neg\varphi$	iff	$I \nvDash \varphi$
$I \vDash \varphi \wedge \psi$	iff	$I \vDash \varphi$ and $I \vDash \psi$
$I \vDash \varphi \vee \psi$	iff	$I \vDash \varphi$ or $I \vDash \psi$
$I \vDash \varphi \Rightarrow \psi$	iff	if $I \vDash \varphi$, then $I \vDash \psi$
$I \vDash \varphi \Leftrightarrow \psi$	iff	$I \vDash \varphi$ if and only if $I \vDash \psi$

If $I \vDash \top$, we say that I satisfies (entails) φ or that I is a model of φ

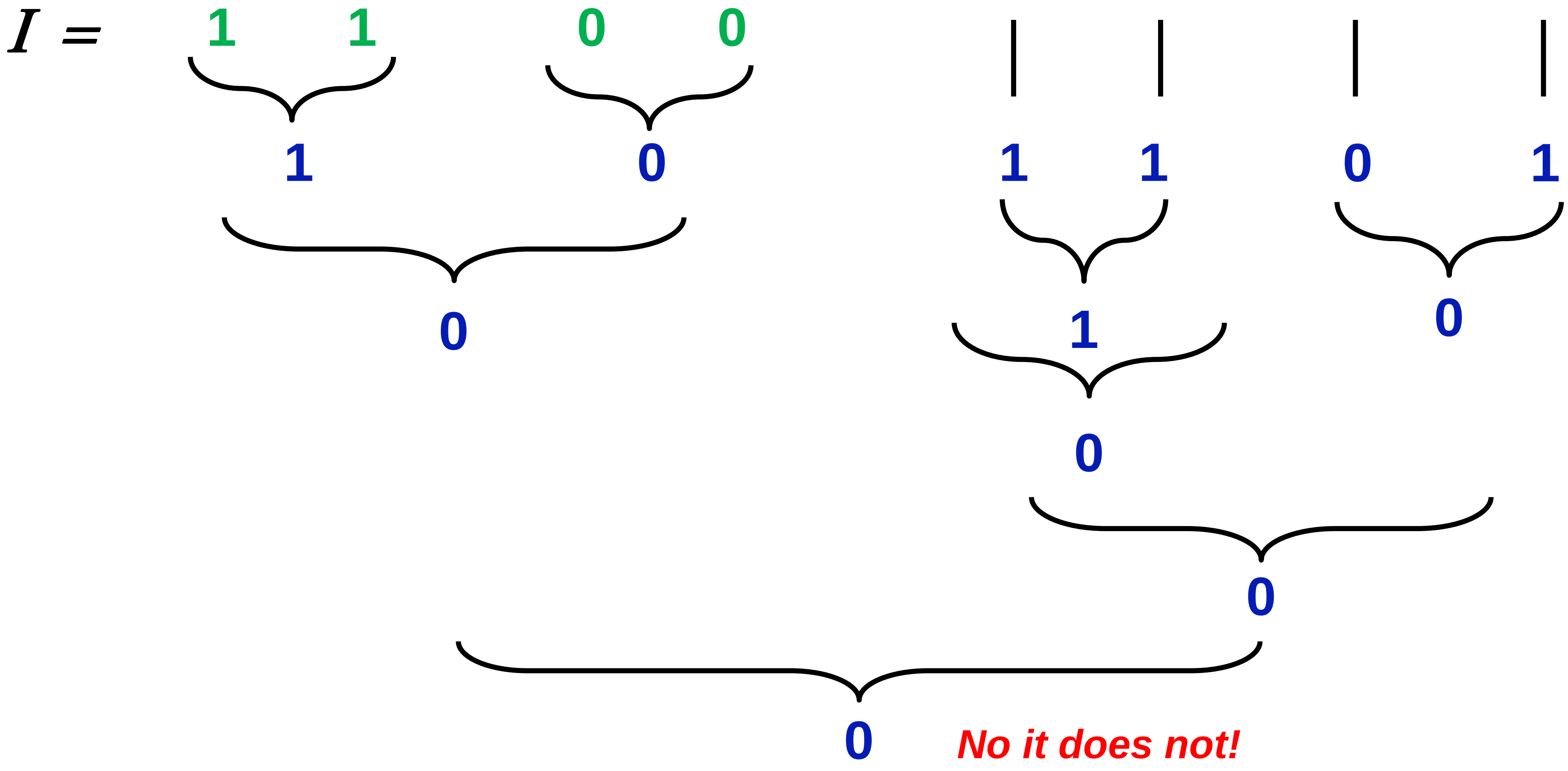
The set of all models (all satisfying interpretations) of φ is denoted by $M(\varphi)$

Alternative Representation: Truth Table for Basic Connectives

φ	ψ	$\neg\varphi$	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\varphi \Rightarrow \psi$	$\varphi \Leftrightarrow \psi$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Example: Does $I \models \varphi$?

$$\varphi = [(P \vee Q) \leftrightarrow (R \vee S)] \wedge [\neg(P \wedge Q) \wedge (R \wedge \neg S)]$$



No it does not!

Terminology

A Knowledge Base (KB) is a set (conjunction) of formulas.

An interpretation is a model of KB if $I \models \varphi$ for all $\varphi \in \mathrm{KB}$.

A formula φ is:

- **satisfiable** iff there exists I that satisfies φ
- **unsatisfiable** iff φ is not satisfiable
- **falsifiable** iff there exists I that doesn't satisfy φ
- **valid** iff $I \models \varphi$ holds for all I. We also call φ a tautology

Formulas φ and ψ are equivalent ($\varphi \equiv \psi$) iff $M(\varphi) = M(\psi)$

Important Equivalences

1. $\neg\neg\varphi \equiv \varphi$
2. $\neg(\varphi \land \psi) \equiv (\neg\varphi \lor \neg\psi)$ (de Morgan)
3. $\neg(\varphi \lor \psi) \equiv (\neg\varphi \land \neg\psi)$ (de Morgan)
4. $(\varphi \Leftrightarrow \psi) \equiv [(\varphi \Rightarrow \psi) \land (\psi \Rightarrow \varphi)]$
5. $(\varphi \Rightarrow \psi) \equiv (\neg\varphi \lor \psi)$
6. $\varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi)$ (distributivity)
7. $\varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi)$ (distributivity)
8. $\varphi \land \neg\varphi \equiv \mathrm{F}$
9. $\varphi \lor \neg\varphi \equiv \mathrm{T}$

Normal Forms

- A formula is in conjunctive normal form (CNF) if it consists of a conjunction of disjunctions of literals:

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

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

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

Example

$$\big((P \lor Q) \land \neg Q\big) \Rightarrow P$$

- In Conjunctive Normal Form and Disjunctive Normal Form

$= \neg\big((P \lor Q) \land \neg Q\big) \lor P$ *(replace implication)*

$= (\neg(P \lor Q) \lor \neg\neg Q) \lor P$ *(move negation inward)*

$= \big((\neg P \land \neg Q) \lor Q\big) \lor P$ *((re-)move negation inward)*

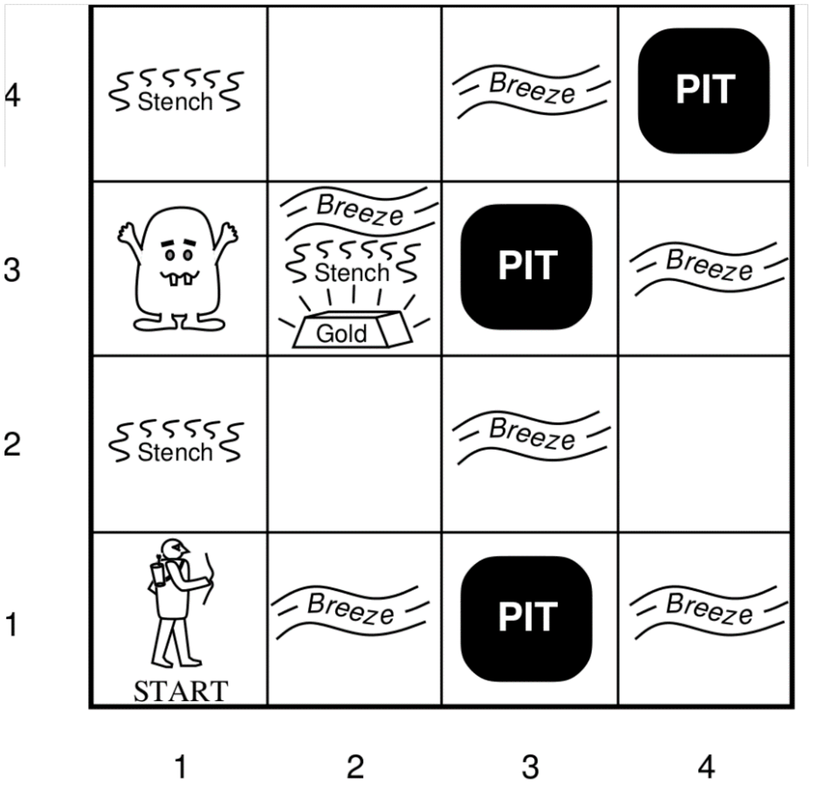
$= (\neg P \land \neg Q) \lor Q \lor P$ DNF

$= \big((\neg P \lor Q) \land (\neg Q \lor Q)\big) \lor P$ *(distribute conjunction)*

$= (\neg P \lor Q \lor P) \land (\neg Q \lor Q \lor P)$ CNF

Can be simplified further to $(\mathrm{T} \lor Q) \land (\mathrm{T} \lor P)$ which is a tautology

Knowledge Base of the Wumpus World



A	= Agent
B	= Breeze
G	= Glitter, Gold
OK	= Safe square
P	= Pit
S	= Stench
V	= Visited
W	= Wumpus

$A11 \land \neg B11 \land \neg S11$

$A11 \rightarrow V11, \ldots$

$S11 \rightarrow W12 \lor W21, S12 \rightarrow \ldots$

$B11 \rightarrow P12 \lor P21, B12 \rightarrow \ldots$

$\neg S11 \rightarrow \neg W12 \land \neg W21, \ldots$

$\neg B11 \rightarrow \neg P12 \land \neg P21, \ldots$

$A11 \land \neg B11 \land \neg S11 \rightarrow OK21 \land OK12?$

Entailment

A knowledge base KB entails a formula φ, $KB \vDash \varphi$, (φ follows from KB) iff φ is true in all models of KB.

$$M\left(\bigwedge_{\psi \in KB}\right) \subseteq M(\varphi)$$

Contradiction Theorem: $KB \vDash \varphi$ if and only if $KB \cup \{\neg\varphi\}$ is unsatisfiable.

Proof:

"$\Longrightarrow$": Assumption $KB \vDash \varphi$

Then for any I where $I \vDash KB$ we have $I \vDash \varphi$ and thus, $I \nvDash \neg\varphi$.

"$\Longleftarrow$": Assumption $KB \cup \{\neg\varphi\} \vdash \bot$

Then for any I where $I \vDash KB$ we have $I \nvDash \neg\varphi$ and thus, $I \vDash \varphi$.

Calculus

- How can we find out $KB \vDash \varphi$?

1. By the truth table method, but n boolean variables require to consider 2^n truth value combinations

2. Using the contradiction theorem, we can try to derive with the help of inference rules (a „calculus“) that

$$KB \cup \{\neg\varphi\} \vDash \perp$$

Inference Rule - Calculus - Deduction

- An inference rule (tautology) specifies how to derive true formulas (logical consequences) from existing true formulas (premises)

  Example: modus ponens

$$(\varphi \wedge (\varphi \Rightarrow \psi)) \Rightarrow \psi, \qquad \frac{\varphi, \varphi \Rightarrow \psi}{\psi}$$

- A calculus is a set of inference rules to perform deductive reasoning (deduction)

- A derivation is a sequence of applied inference rules

Soundness and Completeness of a Calculus

A formula φ can be derived from a knowledge base KB using a calculus $\mathcal{R}$

$$KB \vdash_{\mathcal{R}} \varphi$$

iff there is a derivation using rules from $\mathcal{R}$ ending in φ.

Soundness: $\mathcal{R}$ is sound iff all derivable formulas follow logically:

$$\text{if } KB \vdash_{\mathcal{R}} \varphi \text{ then } KB \vDash \varphi.$$

Completeness: $\mathcal{R}$ is complete if all formulas that follow logically are derivable:

$$\text{if } KB \vDash \varphi, \text{then } KB \vdash_{\mathcal{R}} \varphi.$$

If $\mathcal{R}$ is **sound** and **complete**: $KB \vDash \varphi \Leftrightarrow KB \vdash_{\mathcal{R}} \varphi$.

The Resolution Calculus

- Unit resolution takes a **clause - a** disjunction of literals - and a single literal (the **unit clause**) and produces a new clause (the **resolvent**)

$$\boxed{\dfrac{l_1 \vee \cdots \vee l_k, \quad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}}$$

- The literal l_i and the unit clause m are complementary literals (one is the negation of the other)
- The resolvent contains a single copy of all literals except l_i and m
  - note that a unit clause m remains available in KB for further resolution steps

Generalized Resolution Rule

- We can resolve between non-unit clauses containing complementary literals l_i and m_j
- Resulting clause contains only one copy of each literal

$$\boxed{\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}}$$

- ATTENTION: you can only resolve <u>a single pair</u> of complementary literals in each resolution step

Correct:

$$\frac{(A \vee B) \wedge \neg B}{A} \qquad \frac{(A \vee B \vee P \vee Q) \wedge (\neg P \vee \neg Q)}{(A \vee B \vee P \vee \neg P)}$$

Incorrect:

$$\frac{(A \vee B \vee P \vee Q) \wedge (\neg P \vee \neg Q)}{(A \vee B)}$$

the correct resolvents are: $(A \vee B \vee Q \vee \neg Q)$ and $(A \vee B \vee P \vee \neg P)$! (both are tautologies)

Clause Notation for CNF

- Apply a set-like notation to formulas in CNF

  $(A \lor B \lor P \lor Q) \land (\neg P \lor \neg Q)$

  becomes

  $\{A, B, P, Q\}, \{\neg P, \neg Q\}$ in clause notation

  $(A \lor B \lor P) \land (\neg P)$

  becomes

  $\{A, B, P\}, \{\neg P\}$ in clause notation

  There is no clause notation for DNF (using ; or whatever) 🙄

Example

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

We show by resolution that $KB \cup \{\neg\varphi\} \vDash \perp$

Transform into CNF:

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

We use compact clause notation and add the negation of φ to KB:

$$KB = \{\neg Q, \neg P\}, \{P, \neg Q, \neg R, \neg S\}, \{Q, \neg S\}, \{R, \neg S\}, \{S\}$$
$$KB' = \{\neg Q, \neg P\}, \{P, \neg Q, \neg R\}, \{Q\}, \{R\}, \{S\}$$
$$KB'' = \{\neg P\}, \{P, \neg R\}, \{Q\}, \{R\}, \{S\}$$
$$KB''' = \{\neg P\}, \{P\}, \{Q\}, \{R\}, \{S\}$$

KB'''' $= \perp = \square =$ („$\square$“ stands for the empty clause representing False)

The 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 $false$
      $clauses \leftarrow clauses \cup new$
```

An Unsound and Incorrect Resolution Step

$$KB = \{\neg Q, \neg P\}, \{Q, P\} = \square \text{ ???}$$

- Applying both literals at once yields the empty clause
- However, the KB is satisfiable and not unsatisfiable !!!
- Let us construct a satisfying interpretation for KB

$$KB = \{\neg Q, \neg P\}, \{Q, P\}$$

$$\underbrace{\underbrace{1 \quad 0}_{1} \quad \underbrace{0 \quad 1}_{1}}_{1}$$

$$\{\neg Q, \neg P\}, \{Q, P\}$$

$$\Downarrow$$

two possible resolvents:

$\{\neg P, P\}$ $\{\neg Q, Q\}$

Theorem: Refutation Completeness

KB is unsatisfiable if and only if $KB \vdash \square$.

Proof of Soundness (if):

Follows directly from the contradiction theorem. $(KB \cup \neg\,\square \vDash \bot)$

Proof of Completeness (only if):

It suffices to show that if $KB \nvdash \square$ then KB is satisfiable.

We prove this by induction on $|\Sigma|$ (number of atoms in KB).

We define $RC(KB)$ to be the set of all clauses that can be derived from KB, so $RC(KB) = \{\varphi \mid KB \vdash \quad \}$.

Note, that $KB \nvdash \square$ is equivalent to $\square \notin RC(KB)$, so we assume $\square \notin RC(KB)$ and want to find a valid interpretation of KB.

Base Case: $|\Sigma| = 0$

Since $\square \notin RC(KB)$, $I = \emptyset$ is a valid interpretation of KB, so KB is satisfiable as required.

Inductive Hypothesis: The claim holds for $|\Sigma| = n$.

Inductive Step:

Suppose $|\Sigma| = n + 1$ and $\Sigma = \{P_1, \dots, P_{n+1}\}$. Further, let $\Sigma' = \{P_1, \dots, P_n\}$ and $RC'(KB)$ be the set of all clauses in $RC(KB)$ that do not contain P_{n+1}.

By the induction hypothesis, there exists an interpretation I' such that I' satisfies all clauses in $RC'(KB)$.

We now claim that either $I' \cup \{P_{n+1} \to 0\}$ or $I' \cup \{P_{n+1} \to 1\}$ is a valid interpretation of $RC(KB)$.

For contradiction, suppose none of $I' \cup \{P_{n+1} \to 0\}$ or $I' \cup \{P_{n+1} \to 1\}$ is a valid interpretation of $RC(KB)$.

Then there must be two clauses, $C, D \in RC(KB)$ such that C is not satisfied by $I' \cup \{P_{n+1} \to 0\}$ and D is not satisfied by $I' \cup \{P_{n+1} \to 1\}$.

Since I' is a valid interpretation of $RC'(KB)$, both C and D must contain P_{n+1}. Hence, $P_{n+1} \in C$ and $\neg P_{n+1} \in D$.

Let R be the resolvent of C and D. Then $R \in RC(KB)$ and R does not use P_{n+1}, so I' satisfies R.

As R is satisfied under I', any clause $R \cup X$ (with X being an arbitrary atom or clause) is satisfied by I' independently of the interpretation of X.

Consequently, extending I' to $I' \cup \{P_{n+1} \to 0\}$ or $I' \cup \{P_{n+1} \to 1\}$ satisfy both $R \cup P_{n+1}$ and $R \cup \neg P_{n+1}$ and also satisfies $D, C \in RC(KB)$. This is a contradiction. ■

If $\Box \notin RC(KB)$, we can find a satisfying assignment by greedy value selection as follows:

- Start with an empty interpretation $I = \emptyset$
- From the proof (case $n = 1$):
  - either $\{P_1 \rightarrow 0\}$ or $\{P_1 \rightarrow 1\}$ satisfies the set of all clauses in $RC(KB)$ that do not contain $\{P_2, \dots, P_n\}$
  - So we select the “good” interpretation for P_1 and add it to I
- Once we have selected values for $\{P_1, \dots, P_{k-1}\}$:
  From the proof (case $n = k - 1$):
  - either $I \cup \{P_k \rightarrow 0\}$ or $I \cup \{P_k \rightarrow 1\}$ satisfies the set of all clauses in $RC(KB)$ that do not contain $\{P_{k+1}, \dots, P_n\}$
  - Select the “good” value for P_k and add it to I
- Once we have reached n:
  - the interpretation I we have found, satisfies the set of all clauses in $RC(KB)$, so I is a satisfying assignment

Note that we consider all clauses in the deductive closure of KB, which can have size exponential in the size of the KB. Satisfiability is an NP-hard problem.

Resolution-Based Thinking in the Wumpus World

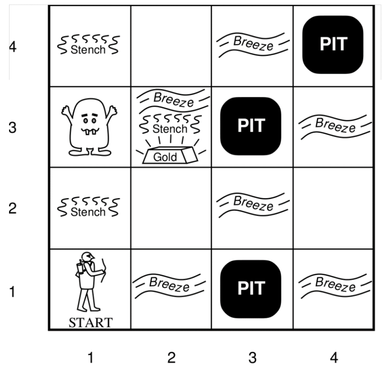
Percepts:

a) $\neg B11$

b) $\neg S11$

Rules:

1) $S11 \rightarrow W12 \vee W21$
2) $B11 \rightarrow P12 \vee P21$
3) $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21$
4) $\neg B11 \rightarrow \neg P11 \wedge \neg P12 \wedge \neg P21$
5) $\neg W12 \wedge \neg P12 \rightarrow OK12$
6) $\neg W21 \wedge \neg P21 \rightarrow OK21$



To show: 7) $OK21 \wedge OK12$

$KB =$

$\{\neg B11\},\{\neg S11\},\{\neg S11, W12, W21\},\{\neg B11, P12, P21\},$	(percepts a,b, rules1,2)
$\{S11, \neg W11\},\{S11, \neg W12\},\{S11, \neg W21\},$	(rule 3 as clause set)
$\{B11, \neg P11\},\{B11, \neg P12\},\{B11, \neg P21\},$	(rule 4 as clause set)
$\{W12, P12, OK12\},\{W21, P21, OK21\},$	(rules 5 and 6 as clause set)
$\{\neg OK12, \neg OK21\}$	(negation of 7 and 8)

rewriting rule 3: $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21 \Leftrightarrow S11 \vee (\neg W11 \wedge \neg W12 \wedge \neg W21)$
$\Leftrightarrow (S11 \vee \neg W11) \wedge (S11 \vee \neg W12) \wedge (S11 \vee \neg W21)$
rewriting rule 5: $\neg W12 \wedge \neg P12 \rightarrow OK12 \Leftrightarrow \neg(W12 \vee P12) \rightarrow OK12$
$\Leftrightarrow (W12 \vee P12) \vee OK12$

Resolution Steps

$\{\neg B11\},\{\neg S11\},\{\neg S11, W12, W21\},\{\neg B11, P12, P21\},$
$\{S11, \neg W11\},\{S11, \neg W12\},\{S11, \neg W21\},$
$\{B11, \neg P11\},\{B11, \neg P12\},\{B11, \neg P21\},$
$\{W12, P12, OK12\},\{W21, P21, OK21\},\{\neg OK12, \neg OK21\}$

Resolution on unit clauses
$\{\neg B11\},\{\neg S11\}$

$\Downarrow$

$\{\neg B11\},\{\neg S11\},\{\neg S11, W12, W21\},\{\neg B11, P12, P21\},$
$\{\neg W11\},\{\neg W12\},\{\neg W21\},$
$\{\neg P11\},\{\neg P12\},\{\neg P21\},$
$\{W12, P12, OK12\},\{W21, P21, OK21\},\{\neg OK12, \neg OK21\}$

Resolution on unit clauses
$\{\neg W12\},\{\neg W21\}$

$\Downarrow$

$\{\neg B11\},\{\neg S11\},\{\neg S11, W12, W21\},\{\neg B11, P12, P21\},$
$\{\neg W11\},\{\neg W12\},\{\neg W21\},$
$\{\neg P11\},\{\neg P12\},\{\neg P21\},$
$\{P12, OK12\},\{P21, OK21\},\{\neg OK12, \neg OK21\}$

Resolution on unit clauses
$\{\neg P12\},\{\neg P21\}$

$\Downarrow$

$\{OK12\},\{OK21\},\{\neg OK12, \neg OK21\}$

Resolution on unit clauses
$\{OK12\},\{OK21\}$
(one of it would be sufficient)

$\Downarrow$

$\square$

Incompleteness of Resolution

$(P \lor Q) \land (\neg Q \lor R) \vDash (P \lor R \lor S)$ is true and we can prove it by showing $(P \lor Q) \land (\neg Q \lor R) \land \neg(P \lor R \lor S) \vdash \square$ by resolution

But we cannot derive $(P \lor R \lor S)$ directly from the sentence using the resolution rule

$\{P, Q\}, \{\neg Q, R\}, \{\neg P\}, \{\neg R\}, \{\neg S\}$

(KB in clause form, negation of claim in clause form)

⬇ resolution on Q

$\{P, R\}, \{\neg P\}, \{\neg R\}, \{\neg S\}$

⬇ resolution on P

$\{R\}, \{\neg R\}, \{\neg S\}$

⬇ resolution on R

$\square$

SAT Checking

- Is an (arbitrary) formula φ satisfiable?
  - Yes, if we can construct a model of φ
  - Find a truth assignment (a satisfying interpretation) for the boolean variables in φ making the formula true
  - If no model exists, return unsatisfiable

- If $\mathrm{K}B \cup \{\neg\varphi\}$ is unsatisfiable, then $\mathrm{K}B \vDash \varphi$ (by the contradiction theorem)
  - SAT checking can be used like a calculus

Characterization of Clauses (Disjunction of Literals)

- **Unsatisfied** if all its literals are assigned value 0 (false)
- **Satisfied** if at least one of its literals is assigned value 1 (true)
- **Unit** if all literals but one are assigned value 0, and the remaining literal is unassigned
  - This remaining literal must be assigned 1
- **Unresolved** if it is neither unsatisfied, nor satisfied, nor unit

Key Observations About Clauses - Early Termination

- A clause is true if *at least one* literal is true
- A conjunction of clauses (a sentence) is false if *at least one* clause is false, which occurs when each of its literals is false
  - even if other literals/clauses do not yet have truth values
  - sentence can be judged false/true before model is completely constructed

$$(A \lor B) \land (A \lor C)$$

is true if A is true, regardless of the values of B and C

➢ ***Early termination*** avoids examination of entire subtrees in the search space when constructing an interpretation

Key Observations about Clauses - Pure Symbol Heuristic

- A boolean variable can occur positively or negatively in a set of clauses
  - Some symbols are „pure" occuring only in one of the forms (with the same „sign")

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

A is pure (only the positive literal appears)

B is pure (only the negative literal appears)

C is impure

- If a sentence has a model, then it has a model with the pure symbols assigned so as to make their literals true, because doing so can never make a clause false (A *true*, $\neg B$ *true*)

➢ ***Pure symbol heuristic*** allows an algorithm to quickly detect true clauses

Key Observations about Clauses - Unit Clause Heuristic

- Generalize the notion of unit clauses from clauses with a single literal to clauses where all but one are assigned *false*
  - The remaining unassigned literal must be *true*

$(\neg B \vee \neg C)$

Simplifies to $\neg C$ with $B = \text{true}$

C must thus be *false* for $\neg C$ to be true

- ➢ Use unit clause heuristic for **unit propagation**
  - Cascade of forced truth value assignments because each unit clause must be *true* for the overall knowledge base (or formula) to be *true*

Davis-Putnam Logemann-Loveland (DPLL) Algorithm

```
Let $F$ be a CNF formula
Let $I$ be an empty assignment

function DPLL($F$, $I$) returns UNSAT or a valid assignment
    if $F$ contains the empty clause then return UNSAT
    if $F$ contains a clause $c$ such that $I(c) = 0$ then return UNSAT
    $(F, I) \leftarrow$ UnitPropagate$(F, I)$
    if $F$ has no clauses left then
        return $I$
    $l \leftarrow$ a literal not assigned by $I$
    if DPLL$(F \setminus \{l\}, I \cup \{I(l) = 1\}) \neq$ UNSAT then
        return DPLL$(F \setminus \{l\}, I \cup \{I(l) = 1\})$
    else
        return DPLL$(F \setminus \{\neg l\}, I \cup \{I(l) = 0\})$

function UnitPropagate($F$, $I$)
    if $F$ has an empty clause then
        return CONFLICT
    while $F$ has a unit clause $x$ but $F \setminus \{x\}$ has no empty clause do
        $F \leftarrow F \setminus \{x\}$
        $I \leftarrow I \cup \{I(x) = 1\}$
    return $(F, I)$
```

splitting rule

Example

unit clause R=*true*

$$KB = \{\{P, Q, \neg R\}, \{\neg P, \neg Q\}, \{R\}, \{P, \neg Q\}\}$$

$\neg R$ is false $\Rightarrow$ remainder of clause must be *true*

unit propagation

$$\{\{P, Q\}, \{\neg P, \neg Q\}, \{P, \neg Q\}\}$$

no pure symbols

Split on P

P=*false*

$\Longrightarrow \{Q\}$ must be *true*

$$\{\{Q\}, \{\neg Q\}\}$$

unit resolution: □

➢ unsatisfying interpretation!

P=*true*

$\Longrightarrow \{\neg Q\}$ must be *true*

$$\{\{\neg Q\}\}$$

Q=*false*

unit propagation: $\mathrm{KB} = \{\}$

➢ satisfying interpretation!

➢ $R, P = T, Q = F$ is a model!

Example

$$\mathrm{K}B = \{\{\neg P, \neg Q\}, \{P, \neg Q, \neg R, \neg S\}, \{R, \neg S\}, \{Q, \neg S\}, \{S\}\}$$

unit propagation: $S = \mathit{true}$

$$\mathrm{K}B = \{\{\neg P, \neg Q\}, \{P, \neg Q, \neg R\}, \{R\}, \{Q\}\}$$

unit propagation: $Q = \mathit{true}$

$$\mathrm{K}B = \{\{\neg P\}, \{P, \neg R\}, \{R\}\}$$

unit propagation: $R = \mathit{true}$

$$\mathrm{K}B = \{\{\neg P\}, \{P\}\}$$
$$= \square$$

➢ unsatisfiable!

Soundness and Incompleteness of Unit Propagation (UP)

- **Soundness**
  - Need to show: If $\mathrm{K}B'$ can be derived from KB by UP, then $\mathrm{K}B \models KB'$
  - **Yes**, because any derivation made by unit propagation can also be made by (full) resolution, which is sound (see Refutation Theorem)
  - (Intuitively: If KB' contains the unit clause $\{l\}$, then l must be made true, so $C \cup \{\bar{l}\}$ implies C)
- **Incompleteness**
  - For completeness need to show: If $KB \models KB'$, then $\mathrm{K}B'$ can be derived from KB by UP
  - **No**, unit propagation only makes limited inferences, as long as there are unit clauses. It does not guarantee to infer everything that can be inferred.
  - Example: $\{\{P, Q\}, \{P, \neg Q\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\}$ is unsatisfiable, but unit propagation cannot derive the empty clause $\square$.

Soundness and Completeness of DPLL (Splitting Rule)

- **Soundness** follows from Soundness of Unit Propagation
- **Completeness**:
  - Without Unit Propagation, DPLL is complete
    - Since we try every possible value assignment in the splitting step
  - Resolution is refutation-complete on CNFs
    - it is guaranteed to derive the empty clause if the given CNF is unsatisfiable
    - If we can deduce the empty clause by applying resolution
      → DPLL deduces unsatisfiable CNF
    - Hence, resolution only “deletes” invalid truth assignments
  - By applying the splitting step we try all truth assignments which have not been shown to be invalid by resolution
    → DPLL is complete

DPLL Exploits Chronological Backtracking

- At each branching step, variable + truth value are selected
  - Two values can be assigned to a variable, either 0 or 1
- Decide on one value and evaluate the logical consequences
- Each time an unsatisfied clause (a conflict) is identified, backtracking is executed
  - undoing branching steps until an unassigned branch is reached
  - when both values have been assigned yielding a conflict, the CNF formula can be declared unsatisfiable

➢ Can we backtrack in a smarter way?

Decision Levels and Antecedents of Variables

$$\varphi = \omega_1 \wedge \omega_2 \wedge \omega_3$$
$$= (x_1 \vee \neg x_4) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee x_2 \vee x_4)$$

Level 1: $x_4 = 0@1$ (choice)
no further assignment through unit propagation

Level 2: $x_1 = 0@2$ (choice)
Unit propagation yields implied assignments
$x_3 = 1@2$ and $x_2 = 1@2$
x_3 is implied by antecedent $\omega_2 : \alpha(x_3) = \omega_2$
x_2 is implied by the antecedent $\omega_3 : \alpha(x_2) = \omega_3$

Implication Graph

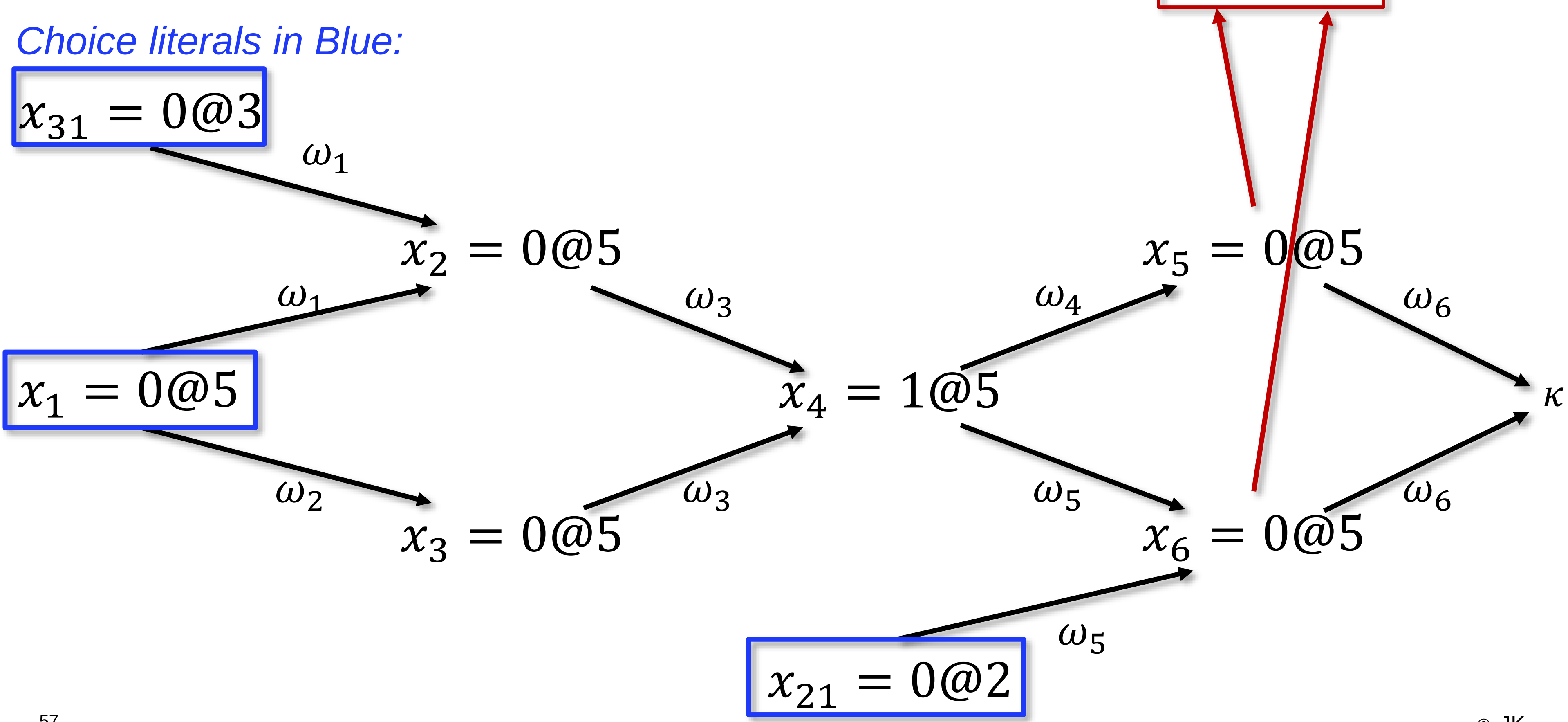
- **Vertices:** all assigned variables and one special node κ (representing the unsatisfied clause)
- **Edges:** obtained from the antecedent of each assigned variable
  - if $\alpha(x_i) = \omega$ then there is a directed edge from each variable in ω, other than x_i, to x_i
  - if unit propagation yields an unsatisfied clause ω_i , then there is a special vertex and a directed edge from ω_i to κ, *i.e.* $\alpha(\kappa) = \omega_i$

$$\begin{tikzcd}
 & x_3 = 1@2 & & x_2 = 1@2 \\
x_1 = 0@2 \arrow[ur, "\omega_2"] \arrow[urrr, "\omega_3"'] & & &
\end{tikzcd}$$

Implication Graph with Conflict

$$\varphi_1 = \omega_1 \land \omega_2 \land \omega_3 \land \omega_4 \land \omega_5 \land \omega_6$$
$$= (x_1 \lor x_{31} \lor \neg x_2) \land (x_1 \lor \neg x_3) \land (x_2 \lor x_3 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (x_{21} \lor \neg x_4 \lor \neg x_6) \land \boxed{(x_5 \lor x_6)}$$

Choice literals in Blue:



Conflict-Driven Clause Learning

- We have decision level $d = 5$ leading to conflict κ
- List of variables implied (by unit propagation) at decision level 5 is: x_5, x_6, x_4, x_2, x_3
- The truth values currently assumed for the choice variables x_{31}, x_1, x_{21} at the root of this implication graph leading to κ are the cause of the conflict

$$\begin{tikzcd}
\boxed{x_{31} = 0@3} \arrow[dr, "\omega_1"] & & & & \\
 & \boxed{x_2 = 0@5} \arrow[dr, "\omega_3"] & & \boxed{x_5 = 0@5} \arrow[dr, "\omega_6"] & \\
\boxed{x_1 = 0@5} \arrow[ur, "\omega_1"] \arrow[dr, "\omega_2"'] & & \boxed{x_4 = 1@5} \arrow[ur, "\omega_4"] \arrow[dr, "\omega_5"'] & & \kappa \\
 & \boxed{x_3 = 0@5} \arrow[ur, "\omega_3"'] & & \boxed{x_6 = 0@5} \arrow[ur, "\omega_6"'] & \\
 & & \boxed{x_{21} = 0@2} \arrow[ur, "\omega_5"'] & &
\end{tikzcd}$$

Learning the Cause of the Conflict

- The assignment of False to these decision variables led to the conflict, so

$$(\neg x_{31} \land \neg x_{21} \land \neg x_1) \Rightarrow \textbf{conflict}$$

- Negating this formula avoids the conflict

$$\begin{aligned} \textbf{no conflict} \quad &\Rightarrow \neg(\neg x_{31} \land \neg x_{21} \land \neg x_1) \\ &\equiv (x_{31} \lor x_{21} \lor x_1) \end{aligned}$$

- So we have learnt that the clause

$$\{x_{31}, x_{21}, x_1\}$$

must be true on our way to a satisfying assignment

Another Example

$$\varphi = \{\{\neg P, \neg Q, R\}, \{\neg P, \neg Q, \neg R\}, \{\neg P, Q, R\}, \{\neg P, Q, \neg R\}\}$$

P is pure!

Apply the Pure Symbol Heuristic and make all occurrences of $\neg P$ true - set P to 0 and we are done!

Let us assume a less smarter choice: $P = 1$, $Q = 1$

$$\varphi = \{\{\underset{0}{\neg P}, \underset{0}{\neg Q}, R\}, \{\underset{0}{\neg P}, \underset{0}{\neg Q}, \neg R\}, \{\underset{0}{\neg P}, \underset{1}{Q}, R\}, \{\underset{0}{\neg P}, \underset{1}{Q}, \neg R\}\}$$

Implied: $R = 1$, $\neg \mathrm{R} = 1$ leading to conflict κ

Learned clause :$\neg(P \wedge Q) \equiv \neg P \vee \neg \mathrm{Q} \equiv \{\neg P, \neg \mathrm{Q}\}$

Conflict-Driven Clause Learning - CDCL Algorithm

```
Let $F$ be a CNF formula
Let $I$ be an empty assignment

function CDCL($F$, $I$) returns UNSAT or a valid assignment
    $d \leftarrow 0$                                                ▷ $d$ is the current decision level
    if UNITPROPAGATE($F$, $I$) = CONFLICT then return UNSAT
    while there are still unassigned variables do
        $x \leftarrow$ a literal or its negation not assigned by $I$          ▷ Branching stage
        $I \leftarrow I \cup \{I(x) = 1\}$
        $d \leftarrow d + 1$                        ▷ Increment decision level due to new decision
        if UNITPROPAGATION($F$, $I$) = CONFLICT then                     ▷ Deduction stage
            blevel $\leftarrow$ CONFICTANALYSIS($F$, $I$)                      ▷ Diagnose stage
            BACKTRACK($F$, $I$, blevel)
            $d \leftarrow$ blevel                  ▷ Decrement decision level due to backtracking
    return $I$
```

- UNITPROPAGATE is as in the DPLL Algorithm
- CONFLICTANALYSIS adds the new learnt clause to KB and returns the backtracking decision level
- BACKTRACK backtracks to the given decision level

Soundness and Completeness of CDCL

- DPLL is a sound and complete algorithm for SAT

- CDCL SAT solvers implement DPLL
  - but can learn new clauses and backtrack nonchronologically

- Clause learning with conflict analysis does not affect soundness or completeness:
  - Conflict analysis identifies new clauses using resolution
  - Hence each learnt clause can be inferred from the original clauses and other learnt clauses by a sequence of resolution steps

Which Other Techniques Enable SAT Solvers to Scale?

- **Component analysis**: when assigning truth values to variables that set of clauses can become separated into disjoint subsets (components) that do not share unassigned variables - the solver can work on each component separately

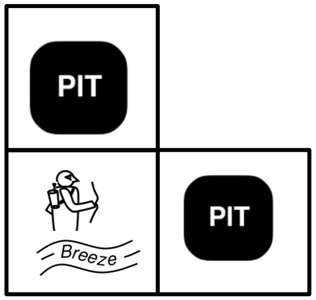
- **Variable and value ordering**: Implement more clever selection of the next variable and the truth value assigment - the degree heuristic suggests to select the variable that appears most frequently over all remaining clauses

- **Clever indexing**: well-thought-trough data structures to find certain variables or clauses quickly

- **Random Restarts**: simply start over when making no progress (see local search lecture), but keep learned clauses

Summary

- Propositional reasoning plays a very important role today in SAT and SMT solvers to tackle large decision, optimization, and verification problems
  – Although boolean variables are very simple and propositional logic seems to be very limited, very complex problems can be represented when using millions of variables or more
- Resolution, DPLL and Clause learning are important algorithms used in modern solvers
- Limits of propositional reasoning result from inherent uncertainty in the domain



  – See figure on the right: the agent cannot obtain further information in cell (1,) if it notices a breeze - a pit can be in (1,2) or (2,1): any move has a 50% risk of death

Working Questions

1. Do you understand the syntax and semantics of propositional logic?
2. What is the essential terminology when talking about syntactic and semantic properties of logical formulas?
3. We learned about two normal forms DNF and CNF: can you transform an arbitrary formula into these normal forms?
4. Can you explain and apply the resolution rule to a set of clauses?
5. What are the main techniques used in DPLL and how do they work?
6. What is the key idea behind conflict-driven clause learning and why does it help a solver to become more effective in finding a satisfiable interpretation or concluding that no such interpretation exists?