```
1  Tell(the initially empty KB, the background knowledge β)
2  repeat  φ ← a sentence describing the current percept π
3          Tell(KB, φ)
4          ψ ←the question "which action should I do now?" as a sentence
5          action ← Ask(KB, ψ)
6          perform this suggested action
7          χ ← a sentence saying "this action has been performed"
8          Tell(KB, χ)
9    until action = "terminate yourself" (if ever)
```

Figure 52: A generic knowledge-based agent. (Russell and Norvig, 2003, Figure 7.1)

Database (e.g. relational) contains explicit *facts*, and answers queries by *fetching* and combining the required facts into results.

E.g. "list the names of all those *student*s who have registered to both *course*s *TEK* and *LAI*".

Knowledge base contains also *sentence*s written in some *knowledge representation language*, and answers queries by *inferring* results from these facts and sentences.

- The idea is to give our agent in Figure 52 a (much) more flexible and powerful kind of memory than e.g. a database.
- We choose the two logics discussed here as our two languages.
- $Tell(KB, \phi)$ adds this sentence ϕ into the knowledge base KB.
- $Ask(KB, \psi)$ asks whether or not this sentence ψ *follows from* the sentences ϕ in KB.

- In the generic knowledge-based agentshown as Figure 52, the

  background knowledge β consists of whatever its designer thinks it must know beforehand. The designer can now express it

  declaratively using the knowledge representation language instead of

  procedurally using some programming language.

  question ψ posed to KB depends on how the *goal* of the agent has been included into its design:

  - Herein Figure 52 we assume that it has been written explicitly as part of the background knowledge β; so we have a goal-based agent as in Section 2.5. Then inference can produce the answer.
  - Our it can be given as a separate utility function instead.
    Then the question ψ posed to KB is "What actions *could* I do now?" instead, and then the utility function selects some *action* from the answer.

4.1 The Wumpus World

- We shall use the *Wumpus world*, as in Figure 53, as our toy running example on using logic to build an intelligent agent.

- Its PEAS description is:

  Performance measure:

  - $+1000$ points for picking up the *gold* — this is the goal of the agent
  - -1000 points for dying = entering a square containing a *pit* or a live *Wumpus* monster
  - -1 point for each action taken, and
  - -10 points for using the *arrow* trying to kill the Wumpus — so that the agent should avoid performing unnecessary actions.

  Environment: A 4×4 grid of squares with...

  - the agent starting from square $[1, 1]$ facing right
  - the gold in one square
  - the initially live Wumpus in one square, from which it never moves
  - maybe pits in some squares.

  The starting square $[1, 1]$ has no Wumpus, no pit, and no gold — so the agent neither dies nor succeeds straight away.

  Actuators: The agent can...

  turn 90° left or right

  walk one square forward in the current direction

  grab an object in this square

  shoot the single arrow in the current direction, which flies in a straight line until it hits a wall or the Wumpus.

  Sensors: The agent has 5 TRUE/FALSE sensors which report a...

  stench when the Wumpus is in an adjacent square — directly, not diagonally

  breeze when an adjacent square has a pit

  glitter when the agent perceives the glitter of the gold in the current square

  bump when the agent walks into an enclosing wall (and then the action had no effect)

  scream when the arrow hits the Wumpus, killing it.

- Let us consider an example gives as Figure 54 on how the agent might reason and act in this Wumpus world.

  Top left: Agent **A** starts at $[1, 1]$ facing right.

  - Agent **A** is actually in the example world given as Figure 53, but does not know it.
  - The background knowledge β assures agent **A** that he is at $[1, 1]$ and that it is **OK** = *certainly* not deadly.
  - Agent **A** gets the percept "**S**tench = **B**reeze = **G**litter = **B**ump = **S**cream = FALSE".
  - Agent **A** *infers* from this percept and β that its both neighbouring squares $[1, 2]$ and $[2, 1]$ are also **OK**: "If *there* was a **P**it (**W**umpus), then *here* would be **B**reeze (**S**mell) — but isn't, so...".
  - The *KB* enables agent **A** to discover certainties about parts of its environment — even without visiting those parts.

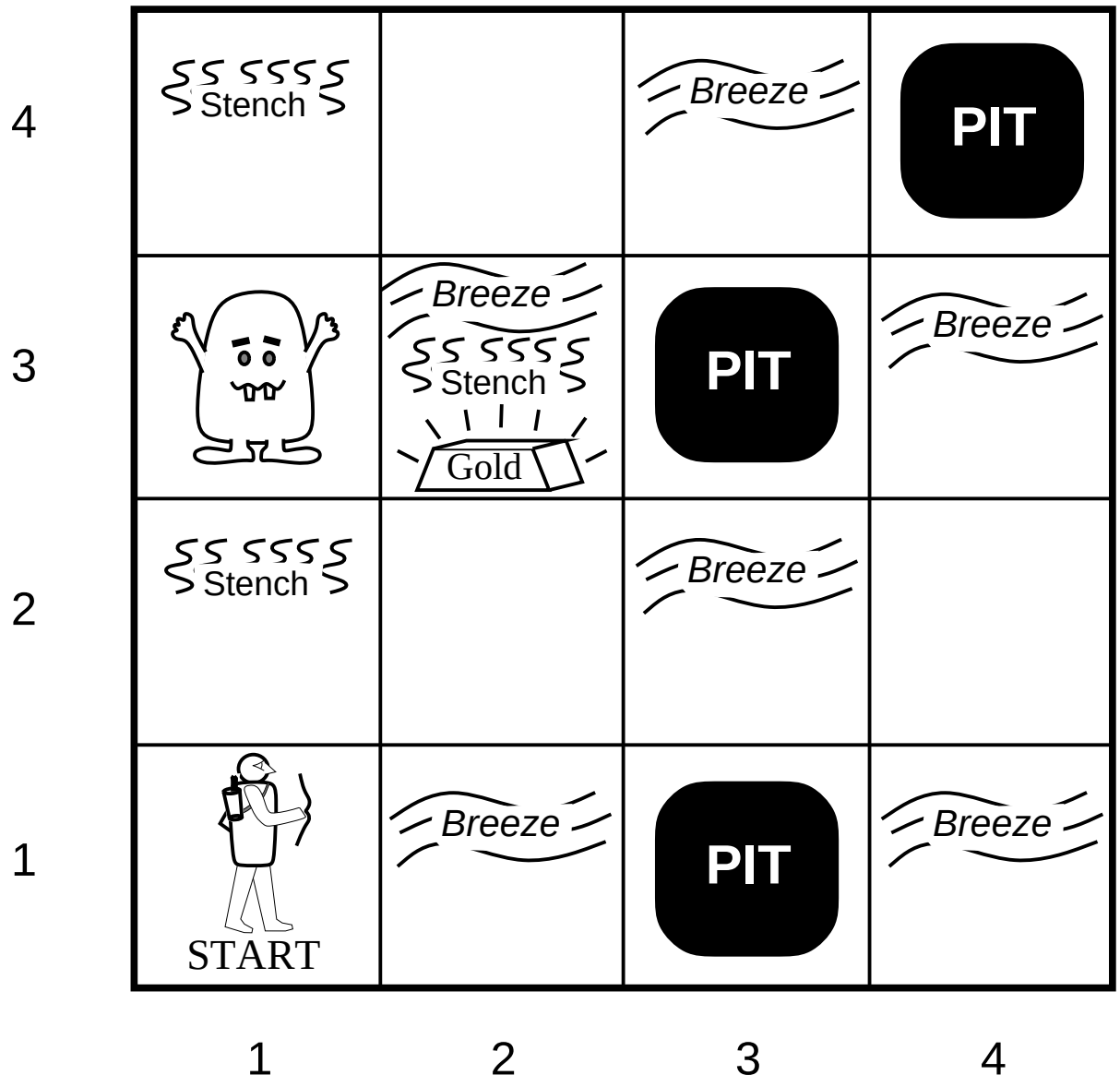

Figure 53: An example Wumpus world. (Russell and Norvig, 2003, Figure 7.2)

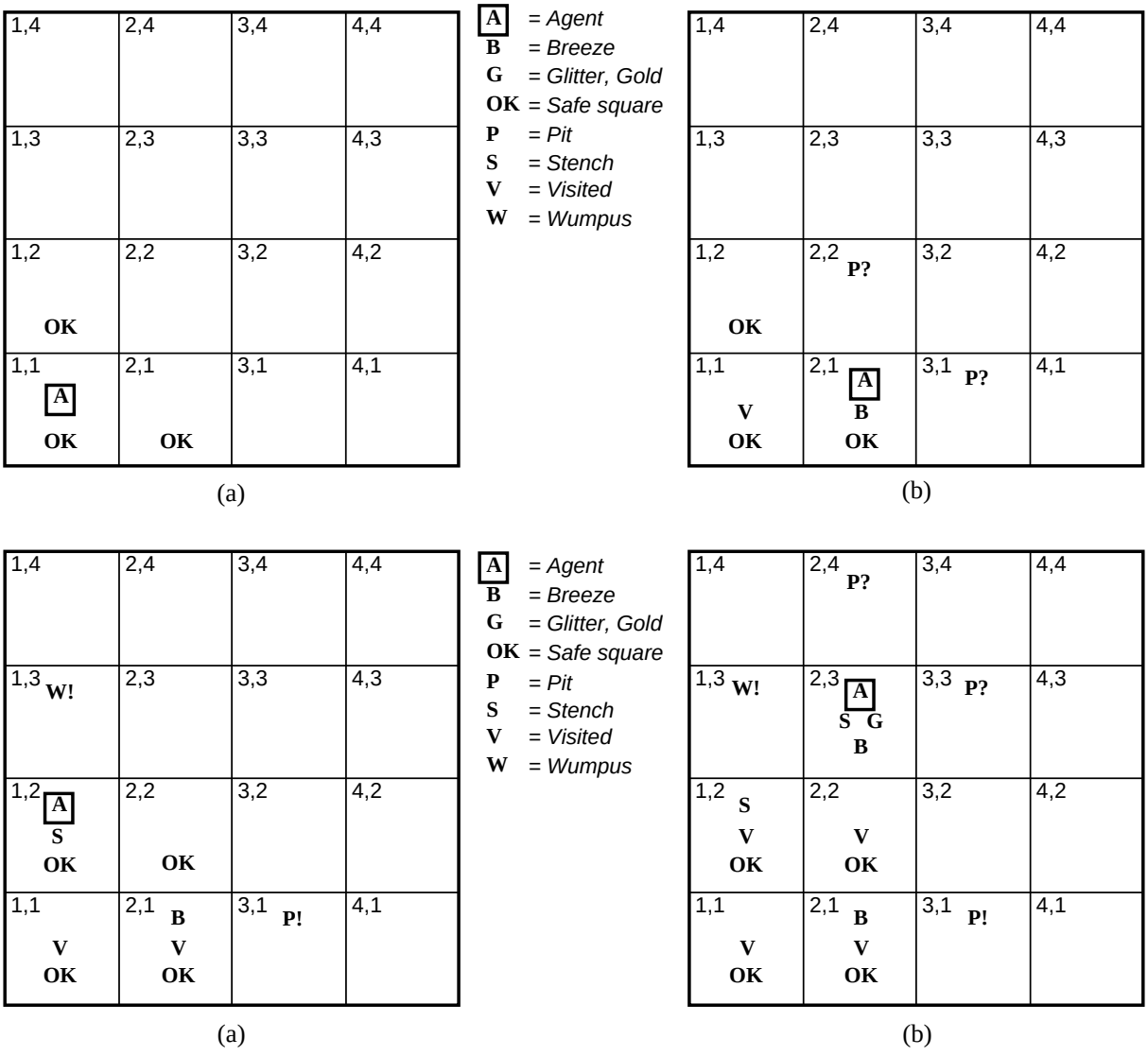

Figure 54: A Wumpus adventure. (Russell and Norvig, 2003, Figures 7.3 and 7.4)

Top right: Agent **A** is cautious, and will only move to **OK** squares.

- Agent **A** walks into $[2, 1]$, because it is **OK**, and in the direction where agent **A** is facing, so it is cheaper than the other choice $[1, 2]$.
  Agent **A** also marks $[1, 1]$ **V**isited.
- Agent **A** perceives a **B**reeze but nothing else.
- Agent **A** infers: "At least one of the adjacent squares $[1, 1]$, $[2, 2]$ and $[3, 1]$ must contain a **P**it. There is no **P**it in $[1, 1]$ by my background knowledge β. Hence $[2, 2]$ or $[3, 1]$ or both must contain a **P**it."
- Hence agent **A** cannot be certain of either $[2, 2]$ or $[3, 1]$, so $[2, 1]$ is a dead end for a cautious agent like **A**.

Bottom left: Agent **A** has turned back from the dead end $[2, 1]$ and walked to examine the other **OK** choice $[1, 2]$ instead.

- Agent **A** preceives a **S**tench but nothing else.
- Agent **A** infers using also *earlier percepts:* "The **W**umpus is in an adjacent square. It is not in $[1, 1]$. It is not in $[2, 2]$ either, because then I would have sensed a **S**tench in $[2, 1]$. Hence it is in $[1, 3]$."
- Agent **A** infers using also *earlier inferences:* "There is no **B**reeze here, so there is no **P**it in any adjacent square. In particular, there is no **P**it in $[2, 2]$ after all. Hence there is a **P**it in $[3, 1]$."
- Agent **A** finally infers: "$[2, 2]$ is **OK** after all — now it is certain that it has neither a **P**it nor the **W**umpus."

This reasoning is too complicated for many animals — but not for the logical agent **A**.

Bottom right:

1. Agent **A** walks to the only unvisited **OK** choice $[2, 2]$. There is no **B**reeze here, and since the the square of the **W**umpus is now known too, $[2, 3]$ and $[3, 2]$ are **OK** too.
2. Agent **A** walks into $[2, 3]$ and senses the **G**litter there, so he grabs the gold and succeeds.

4.2 Principles behind Logic(s)

- Let us now review the fundamental principles behind logic(s) from our intelligent agent viewpoint.
- From this viewpoint, the actual *syntax* in which the logical *statement*s are written as *formula*s is just a "print-out" of the underlying data structure.
  - This underlying data structure is the parse tree of the *formula*.
  - From a programming viewpoint, implementations of logical inference create from existing such structures new ones in ways which respect their *meaning*.
- This meaning or *semantics* of a *statement* defines whether it is TRUE or FALSE in the given *possible world*.
  - In our agent viewpoint, possible world = possible state of the environment.

– Formal semantics for logic(s) uses the term *interpretation* or *model* = a possible world $w+$ a "dictionary" telling how the different parts of the *statement* are to be understood in w.

We rarely need to make this distinction.

(The meaning of a modal logic involves not just one but several possible worlds — the different viewpoints.)

- Then a given *statement* ϕ stands for the set of all those possible worlds w such that ϕ is TRUE in w.

  Formal semantics calls these w the *models* of ϕ and denotes this set as $\text{Mod}(\phi)$.

- E.g. $\text{Mod}(KB)$ after agent **A** has walked into $[2,1]$ and perceived only the **B**reeze there is shown in Figure 55 and consists of all the worlds such that
  - there is **B**reeze in $[2,1]$, and
  - all the possible **P**it placements around it, according to his background knowledge β, but
  - the other squares of the worlds are not drawn, because agent **A** does not know anything more than β about them (yet).

- Here KB is considered to be the combined *statement*

$$\beta \textbf{ and } \alpha_1 \textbf{ and } \alpha_2 \textbf{ and } \alpha_3 \textbf{ and} \ldots \textbf{and } \alpha_t$$

  where these α_i are the individual *statement*s that have been added into KB using tell *Tell* until now.

  Formal semantics often considers the *theory* $\{\beta, \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_t\}$ instead, to permit also infinite theories.

- The central semantic relation between two *statement*s is

$$\gamma \textit{ entails } \delta$$

  which is written as

$$\gamma \models \delta$$

  and means that "**if** γ is TRUE, **then** δ must also be TRUE". For worlds, this is the same thing as

$$\text{Mod}(\gamma) \subseteq \text{Mod}(\delta).$$

  E.g.

$$KB \models \alpha_1$$

  for the *statement*

$$\alpha_1 = \text{"}[1,2] \text{ has no } \textbf{P}\text{it"}$$

  in Figure 55. Figure 56 shows another example where KB does *not* entail α_2.

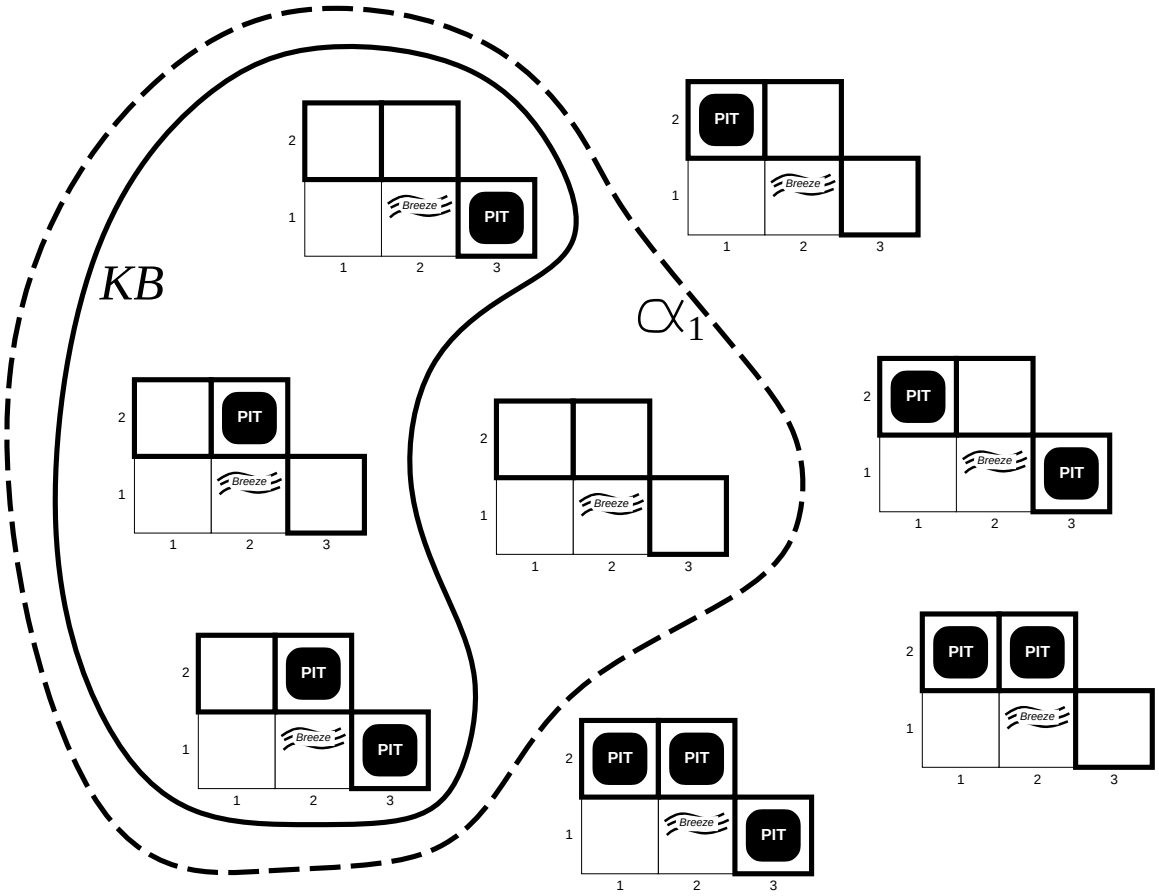

Figure 55: The possible worlds according to KB and an entailed α_1. (Russell and Norvig, 2003, Figure 7.5(a))

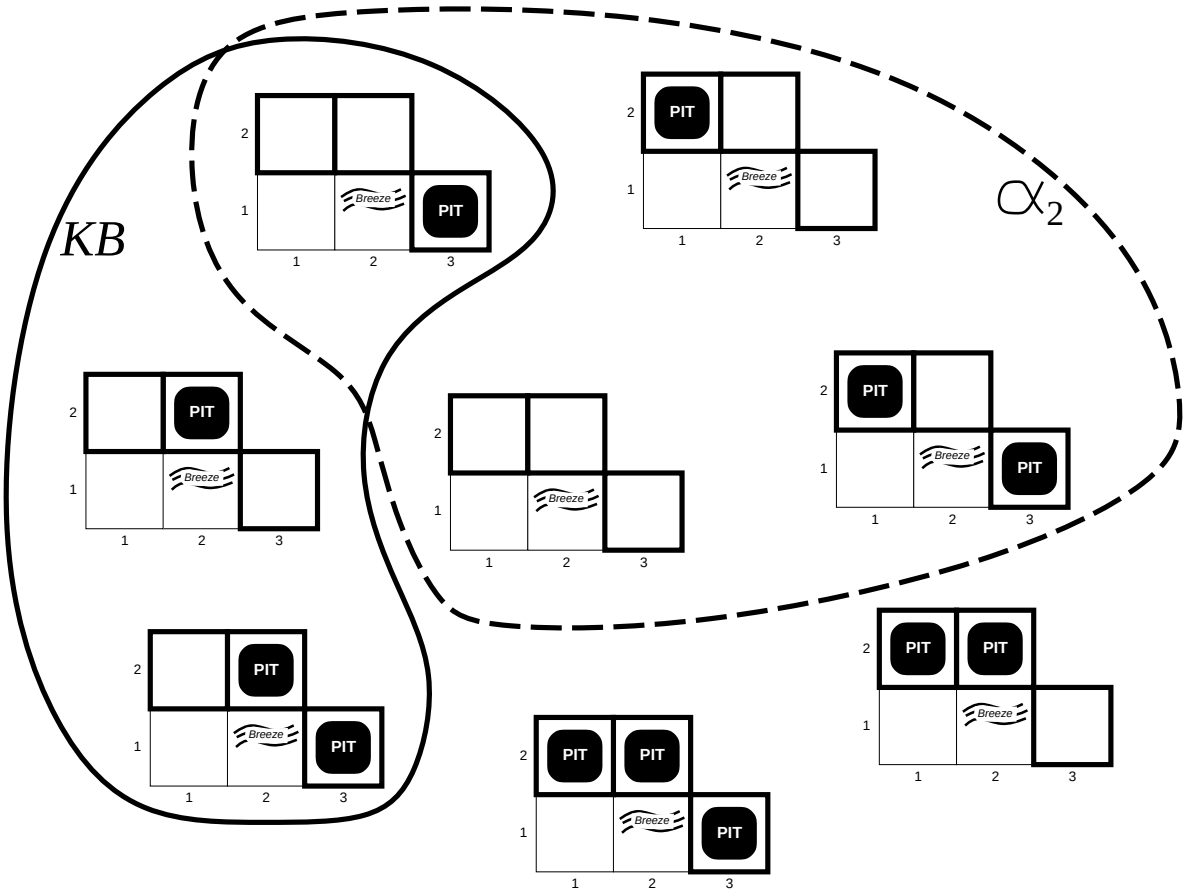

Figure 56: $KB \not\models \alpha_2$. (Russell and Norvig, 2003, Figure 7.5(b))

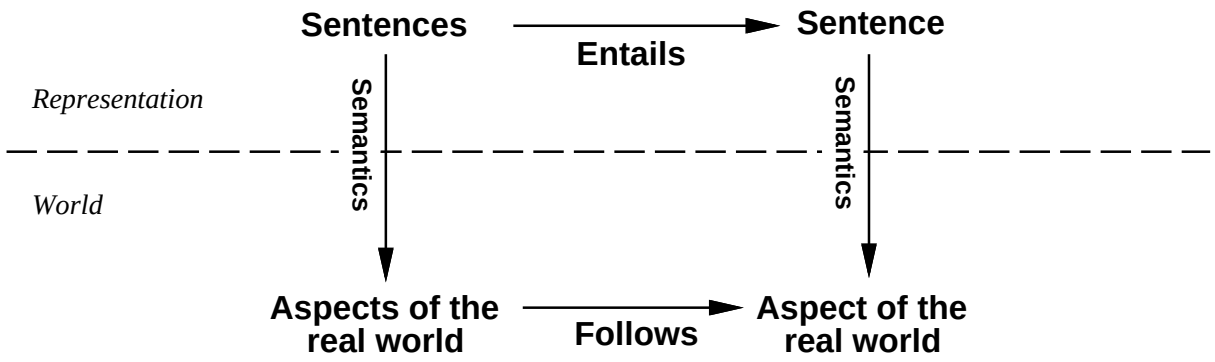

Figure 57: The world and its representation. (Russell and Norvig, 2003, Figure 7.6)

- These sets $\mathrm{Mod}(\phi)$ are the same *belief* sets B_ϕ as in the agent's belief state space $\mathcal{B}$.
  - Conversely, ϕ is one logical representation for this belief set $B_\phi \in \mathcal{B}$.
  - The agent believes that the actual world w_{actual} is one of the worlds $w \in \mathrm{Mod}(KB)$, according to the information KB it now has about w_{actual}.
  - The other worlds $w' \in \mathrm{Mod}(KB)$ than w_{actual} represent the incompleteness of this information, since some things are different in these other w' than in w_{actual}.
  - The philosophical problem of *grounding* is "Why would it be the case that $w_{\text{actual}} \in \mathrm{Mod}(KB)$?"

    The agent approach assumes that its sensors create the connection.
- If you think it is strange that the agent's *knowledge* is a *belief* set, then recall the philosopher Plato's ("Platon" in Finnish) (Greece, 429-4347 BCE) dictum

  "Knowledge is true justified belief".

- In summary, as Figure 57 shows:
  - The *sentence*s in the KB represent various (assumed to be) TRUE *aspect*s of the real world; The *semantics* explains how.
  - Then $KB \models \psi$ ensures that the *aspect* represented by ψ is also TRUE.
- $Ask(KB, \psi)$ on Step 5 of the knowledge-based agent in Figure 52 is to first

  verify that $KB \models \psi$

  provide some appropriate *answer* explaining why.

  This agent view is constructive, even when it uses a classical logic: a blunt "because it just is" would not be a sufficient *answer* for action.
- Hence we need a way to *compute*

  $$KB \models \psi$$

  and an *answer* by manipulating the data structures for KB and ψ instead of considering the possible worlds w.

  There are just too many w to consider — in general, even infitely many!

- We say "statement ψ can be *derived* (or *inferred*) from KB by the *proof system* $\mathcal{P}$", in symbols

$$KB \vdash_{\mathcal{P}} \psi$$

when this computation can be done using a certain specialized kind of search problem $\mathcal{P}$ not discussed yet.

- For this search problem $\mathcal{P}$...

soundness is the *crucial* logical property that $\mathcal{P}$ must not lie:

$$\text{if } KB \vdash_{\mathcal{P}} \psi \text{ then } KB \models \psi.$$

completeness is the opposite *desirable* logical property that everything which is TRUE should also be derivable by $\mathcal{P}$:

$$\text{if } KB \models \psi \text{ then } KB \vdash_{\mathcal{P}} \psi.$$

effectiveness is the desirable *computational* property that the the search "is $KB \vdash_{\mathcal{P}} \psi$?" always terminates with a YES/NO answer.

- An early (1931) celebrated deep logical result is Kurt *Gödel's 1st incompleteness theorem:*

> There cannot be any such proof system $\mathcal{N}$ with all these three properties for the natural numbers $\mathbb{N}$.

4.3 Propositional Logic

- The smallest "addressable unit" of propositional logic ("propositio- eli lauselogiikka" in Finnish) is a *whole sentence* which can be either TRUE or FALSE.

- That is, any expression η such that

$$\text{"Is it the case that } \eta\text{?"}$$

is a meaningful question with a YES/NO answer.

- E.g. asking this about $\eta =$

$$\text{Romeo loves Juliet.} \tag{23}$$

is meaningful, so it is a sentence in this sense.

- On the other hand, e.g. neither "Is it the case that loves?" nor "Is it the case that Juliet?" is a meaningful question.

  We shall later present another logic (the *predicate* logic) where we can address also such parts of sentences.

- Note also that "Is it the case that Romeo loves?" is another meaningful question — but this is a different sentence than Eq. (23).

- The syntax used in this course and its book Russell and Norvig (2003, Figure 7.7) for propositional logic is:

$$\begin{aligned} \mathit{Sentence} &\to \mathit{Atomic} \quad | \quad \mathit{Complex} \\ \mathit{Atomic} &\to \textsc{true} \quad | \quad \textsc{false} \quad | \quad \mathit{Symbol} \\ \mathit{Complex} &\to \neg\, \mathit{Sentence} \\ &\quad | \quad (\mathit{Sentence} \wedge \mathit{Sentence}) \\ &\quad | \quad (\mathit{Sentence} \vee \mathit{Sentence}) \\ &\quad | \quad (\mathit{Sentence} \Rightarrow \mathit{Sentence}) \\ &\quad | \quad (\mathit{Sentence} \Leftrightarrow \mathit{Sentence}) \end{aligned}$$

  - The predicate *Symbol*s are variable names which stand for different *sentence*s. E.g. we might choose the name X to stand for Eq. (23) when we are writing the background knowledge β, and so on.
  - We suppress writing nested parentheses by stipulating that the *connective*s $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ are in descending binding power.
  - We also stipulate that $\wedge, \vee$ are associative, but $\Rightarrow, \Leftrightarrow$ are not.

- In the semantics, a possible world w is just a function

$$w\colon\ \mathit{Symbol} \mapsto \underbrace{\{\textsc{true}, \textsc{false}\}}_{\text{call this } \mathbb{B}}$$

  which gives for each *Symbol* Y its truth value $w(Y)$.

- This w extends from single *Symbol*s to whole *Sentence*s:

$$w(c) = c \text{ if } c \in \mathbb{B}$$

$$w(\neg\phi) = \begin{cases} \textsc{false} & \text{if } w(\phi) = \textsc{true} \\ \textsc{true} & \text{if } w(\phi) = \textsc{false} \end{cases}$$

$$w(\phi \oplus \psi) = \text{look it up from the right line and column from}$$

line		column $\oplus$			
$w(\phi)$	$w(\psi)$	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

  These are the *truth tables* for our binary connectives (Russell and Norvig, 2003, Figure 7.8).

- If we think that $\textsc{false} < \textsc{true}$, then

  '$\wedge$' is 'min'.

  '$\vee$' is 'max'.

  '$\Rightarrow$' is '$\leq$'.

  This $\phi \Rightarrow \psi$ is *material* implication: the *antecedent* ϕ does not have to be in any way "relevant" or "a cause" for the *consequent* ψ.

  (Philosophers have developed so-called relevance logics which try to capture the idea that ϕ should be somehow relevant to ψ.)

'$\Leftrightarrow$' is '='.

(This connection between logical proofs and calculating with truth values is behind the *calculational proofs* approach, which is handy in Computer Science, e.g. in formal program construction (Backhouse, 2003).)

- The part of the background knowledge β for our Wumpus world which deals with pits can be written in propositional logic as follows:
  - Let the *Symbol* $P_{i,j}$ (or $B_{i,j}$) stand for "There is a **P**it (or **B**reeze) in square $[i, j]$".
  - "There is no pit in $[1, 1]$" is then simply
  $$\neg P_{1,1}. \tag{24}$$
  - We have the general rule that
  > "this square has a **B**reeze *exactly when* at least one of its adjacent squares has a **P**it". (25)
  - But now we encounter a shortcoming of propositional logic:
    * Rule (25) combines together two whole *sentence*s with *"exactly when"*. This we *can* do with '$\Leftrightarrow$'.
    * But rule (25) is trying to combine them so that "this" and "it" are the same square.
    Alas, propositional logic does not allow us to access such parts of whole *statement*s.
  - Since our Wumpus worlds are finite (4×4-square grids), we can get around this problem with a sledgehammer solution: Write a separate copy of rule (25) for each different "this" square. E.g.
  $$\text{corner square: } B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1} \tag{26}$$
  $$\text{edge square: } B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1} \tag{27}$$
  $$\text{other square: } B_{2,2} \Leftrightarrow P_{1,2} \vee P_{2,1} \vee P_{2,3} \vee P_{3,2}. \tag{28}$$
  All these *sentence*s go into the background knowledge β, and they are TRUE in all Wumpus worlds.
  - Then we add into KB also the **B**reeze percepts at the top right Wumpus world of Figure 54:
  $$\neg B_{1,1} \tag{29}$$
  $$B_{2,1}. \tag{30}$$

4.3.1 Propositional Inference

- The aim is to verify that $KB \models \alpha$ by computing $KB \vdash_{\mathcal{P}} \alpha$ using some sound proof system $\mathcal{P}$.
- For our classical propositional logic, we can even select this $\mathcal{P}$ to be also complete and effective.
- Here we take this $\mathcal{P}$ to be some program which the agent is running, and consider some approaches to writing such a program.

- We say that a formula α is...

  satisfiable if α is TRUE in *at least one* possible world w.

  - The computational problem "Is this α satisfiable?" is called the *SATisfiability* problem.
  - For our classical propositional logic, SAT is "the mother of all **NP**-complete problems".
  - As a search problem, it is "try to assign TRUE or FALSE to the *Symbol*s in α in such a way that α becomes TRUE".

  valid if α is TRUE in *every* possible world w. Then this α is...

  - TRUE solely because of its own logical form, regardless of how the world(s) happen to be. E.g. the formalization

    $$\phi \vee \neg\phi$$

    of Eq. (22) is classically valid, regardless of what ϕ "really says" and whether what it says is TRUE or FALSE about the agent's current environment.
  - also called a *tautology*.

- The *deduction theorem* connects entailment and validity:

  $$\gamma \models \delta \text{ if and only if } \gamma \Rightarrow \delta \text{ is valid.} \tag{31}$$

  - That is, the implication '$\Rightarrow$' is at the *language* level the same concept as entailment '$\models$' is at its *semantic* level.
  - It is a starting point for developing new logics: the concept "entails" can be understood in several ways (such as "γ must be somehow relevant to δ") — what kind of "implication" arises if we understand it like this?
  - It also means that we get our program $\mathcal{P}$ for answering the question "$\gamma \models \delta$?" if we develop an algorithm for the question "Is $\gamma \Rightarrow \delta$ valid?".
  - This question can in turn be answered by looking only at the logical form of this formula $\gamma \Rightarrow \delta$.

- We also get the principle of proof by *refutation* or *contradiction* (or *reductio ad absurdum*):

  $$\gamma \models \delta \text{ if and only if } \gamma \wedge \neg\delta \text{ is not satisfiable.} \tag{32}$$

  That is: "If δ was FALSE, then this would contradict γ" (= our KB, which we maintain satisfiable).

- For propositional logic, this question "Is $\gamma \Rightarrow \delta$ valid?" is in turn the *complement* of a search problem:

  - "Try to assign values TRUE or FALSE to the *Symbol*s in $\gamma \Rightarrow \delta$ so that $\gamma \Rightarrow \delta$ becomes FALSE. If there are *no* such values, then $\gamma \Rightarrow \delta$ is valid."
  - Classical propositional validity is **co-NP**-complete.

- The brute-force method shown as Figure 58 for solving SAT is to start

  computing its truth table

= **trying out** all different assignments of TRUE or FALSE into the *Symbol*s appearing in the given *formula* — the other *Symbol*s not in it don't need any value

and stopping with the answer YES if we get the *formula* to be TRUE.

- Validity can be solved in the same way, but stopping with NO if we get the *formula* to be FALSE.

- This method collects into an initially empty data structure v the values currently assigned to *Symbol*s.
  - If a symbol X has not been assigned any value yet, then $v[X] = \bot$.
  - Then $RowValue(\alpha, v)$ = the which the given *formula* α gets with these currently assigned values $v[Y]$ — but not all *Symbol*s Y have been assigned a value yet.
  - Hence we must permit this '$\bot$' also in our truth tables:

		$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
FALSE	*other*	FALSE	*other*	TRUE	$\overline{other}$
other	FALSE	FALSE	*other*	$\overline{other}$	$\overline{other}$
TRUE	*other*	*other*	TRUE	*other*	*other*
other	TRUE	*other*	TRUE	TRUE	*other*

(33)

Here $\overline{other}$ means "flip the *other* value around if it is defined":

$$\overline{\text{TRUE}} = \text{FALSE}$$
$$\overline{\text{FALSE}} = \text{TRUE}$$
$$\overline{\bot} = \bot.$$

- This brute-force method takes up to

$$\mathcal{O}(m \cdot 2^n) \text{ steps, where}$$
$$m = \text{the size of the (parse tree for) formula } \alpha$$
$$n = \text{the number of distinct } \textit{Symbol}\text{s in } \alpha$$

simply because it might try all the 2^n distinct assignments v to the *Symbol*s in α.

- Fortunately we do have algorithms, which take much less in practice.

- However, there are no SAT algorithms which could always guarantee a YES/NO answer in less than exponentially many steps in n (unless $\mathbf{P} = \mathbf{NP}$).

- One way to present a proof system $\mathcal{P}$ is as a collection of *inference rule*s, like the following *Modus Ponens:*

$$\frac{\alpha \quad \alpha \Rightarrow \beta}{\beta} \tag{34}$$

Premisses above the line list all that must hold before this rule can be applied.

Conclusion below the line gives what can then be inferred.

- An inference rule like Eq. (34) can be read in two directions.

TruthTable(α, v)
1 $r \leftarrow$ *RowValue*(α, v)
2 **if** $r = \bot$
3 **then** $X \leftarrow$ some *Symbol* in α such that $v[X] = \bot$
4 **return** *TruthTable*$(\alpha, v$ extended with $v[X] \leftarrow$ TRUE$)$ **or**
TruthTable$(\alpha, v$ extended with $v[X] \leftarrow$ FALSE$)$
5 **else return** r

RowValue(β, v)
1 **if** β is a *Symbol*
2 **then return** $v[\beta]$
3 **if** β is of the form $\neg\phi$
4 **then return** $\overline{\textit{RowValue}(\phi, v)}$
5 **if** β is of the form $\phi \otimes \psi$
6 **then return** the entry for *RowValue*(ϕ, v), *RowValue*(ψ, v) and '$\otimes$' in Eq. (33)
7 **return** β (since $\beta \in \mathbb{B}$ must now be the case)

Figure 58: SAT via truth-table enumeration. Modified from Russell and Norvig (2003, Figure 7.10).

Forward direction reads "if I have already inferred these premisses, then I can infer this conclusion too".

Backward direction reads "I want to infer this conclusion, so I will try to infer these premisses next".

This gives two opposite approaches to computing $KB \vdash_{\mathcal{P}} \beta$.

- Soundness of $\mathcal{P}$ requires that its inference rules like Eq. (34) satisfy the condition

$$\text{the conjunction of all its premisses} \models \text{its conclusion.} \tag{35}$$

This can in turn be checked (e.g.) by computing the corresponding truth table.

- E.g. our agent could infer as follows in the initial top left situation of the Wumpus world shown as Figure 54:

$$\dfrac{\text{percept Eq. (29)} \qquad \dfrac{\dfrac{\dfrac{\text{background Eq. (26)}}{(B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1} \Rightarrow B_{1,1})}\ \Leftrightarrow\text{-elimination}}{P_{1,2} \vee P_{2,1} \Rightarrow B_{1,1}}\ \wedge\text{-elimination}}{\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})}\ \text{contraposition}}{\dfrac{\neg(P_{1,2} \vee P_{2,1})}{\neg P_{1,2} \wedge \neg P_{2,1}}\ \text{de Morgan}}\ \text{Eq. (34)} \tag{36}$$

Here the agent used the following inference rules:

$\Leftrightarrow$-elimination or the definition of '$\Leftrightarrow$' as '$\Rightarrow$' in both directions

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \tag{37}$$

$\wedge$-elimination on the right

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \wedge \beta}{\beta}$$

contraposition or flipping '$\Rightarrow$' around with '$\neg$'

$$\frac{\alpha \Rightarrow \beta}{\neg\beta \Rightarrow \neg\alpha}$$

de Morgan for moving '$\neg$' in and out of '$\wedge$' or $\vee$'

$$\frac{\neg \left(\alpha \begin{array}{c} \vee \\ \wedge \end{array} \beta \right)}{(\neg\alpha) \begin{array}{c} \wedge \\ \vee \end{array} (\neg\beta)} \tag{38}$$

(and vice versa).

- Searching for a proof can be much faster that computing the truth table, because the search can ignore much of *KB* as irrelevant for this current task.

  E.g. here all knowledge about the **W**umpus was ignored, because this current task was to reason about **P**its only.

4.3.2 Propositional Resolution

- Using a proof system $\mathcal{P}$ with many rules is fine for humans.
- For a computer, the less rules $\mathcal{P}$ has the simpler the corresponding search algorithm gets.
- Just *one* rule called *resolution* turns out to be enough — if we
  1. first rewrite our *KB* into a certain fixed form
  2. then seek refutation proofs, as in Eq. (32).
- The general idea of resolution is

$$\frac{\alpha \vee \beta \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

  or "if the same disjunct β occurrs both

  positively (= outside '$\neg$') in one disjunction and
  negatively (= inside '$\neg$') in another disjunction

  then you can combine these two disjunctions into one *resolvent* and drop their common β".

- E.g. our agent could infer as follows in the bottom left situation of the Wumpus world shown as Figure 54:

$$\dfrac{\dfrac{\dfrac{B_{2,1} \quad B_{2,1} \Rightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1} \text{ like in Eq. (36)}}{P_{1,1} \vee P_{2,2} \vee P_{3,1}} \qquad \neg P_{1,1}}{P_{2,2} \vee P_{3,1}}\ \text{resolution} \qquad \neg P_{2,2} \text{ like in Eq. (36)}}{P_{3,1}}\ \text{resolution}$$

The associativity and commutativity of '$\vee$' lets us pick any disjunct in these resolution steps.

- In both of these two resolution steps, the common disjunct β is just a single *Symbol*.
  - We are going to rewrite our *KB* so that this will always be the case.
  - The reason is that an unmodified *KB* would not contain very many possible choices for β.
- The form into which we are going to rewrite our *KB* can be defined as follows:

  Literal is *Symbol* or $\neg$ *Symbol*.

  Clause is a (possibly empty) disjunction of literals.

  Then our *KB* is in *Conjunctive Normal Form (CNF)* if it is a conjunction of clauses.
- The full resolution rule takes the form "you can form the resolvent of two clauses if the same *Symbol* X occurrs positively in one of them and negatively in the other":

$$\frac{\overbrace{p_1 \vee p_2 \vee p_3 \vee \cdots \vee p_m \vee X}^{\text{one clause}} \quad \overbrace{\neg X \vee q_1 \vee q_2 \vee q_3 \vee \cdots \vee q_n}^{\text{the other clause}}}{p_1 \vee p_2 \vee p_3 \vee \cdots \vee p_m \vee q_1 \vee q_2 \vee q_3 \vee \cdots \vee q_n}$$

  By the associativity and commutativity of '$\vee$' this common symbol X can appear anywhere inside these two clauses.
- We must also *factor* the resolvent: if some literal would appear more than once, only one copy is retained.
- Or alternatively we can think that a clause is a *set* of literals:

$$\frac{\{p_1, p_2, p_3, \ldots, p_m, X\} \quad \{\neg X, q_1, q_2, q_3, \ldots, q_n\}}{\{p_1, p_2, p_3, \ldots, p_m, q_1, q_2, q_3, \ldots, q_n\}}$$

  This form includes the factoring rule implicitly.
- A given *formula* can be converted into CNF with the following 4 steps:
  1. Replace each occurrence of '$\Leftrightarrow$' with the corresponding two occurrences of '$\Rightarrow$' as in Eq. (37).
     E.g. Eq. (26) of our Wumpus world background knowledge becomes
     $$(B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1} \Rightarrow B_{1,1}). \tag{39}$$
  2. Replace each occurrence of $\alpha \Rightarrow \beta$ with the (classically) equivalent $\neg\alpha \vee \beta$.
     E.g. Eq. (39) becomes
     $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}). \tag{40}$$
  3. Move each '$\neg$' towards the *Symbol*s using Eq. (38). If you get a double negation like $\neg\neg\alpha$, then erase them, leaving only α.
     E.g. Eq. (40) becomes
     $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}). \tag{41}$$
  4. Finally move each '$\wedge$' from under any '$\vee$' by using their distibutivity, which permits replacing $\alpha \vee (\beta \wedge \gamma)$ with $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ and so on.
     E.g. Eq. (41) becomes
     $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}). \tag{42}$$
     If we now undo step 2, then we see that Eq. 42 does indeed say the same thing as the original Eq. (26), but in a different way:
     $$(B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \Rightarrow B_{1,1}) \wedge (P_{2,1} \Rightarrow B_{1,1}).$$