Grundlagen der Künstlichen Intelligenz

6 Logische Agenten

Rationales Denken, Logik, Resolution

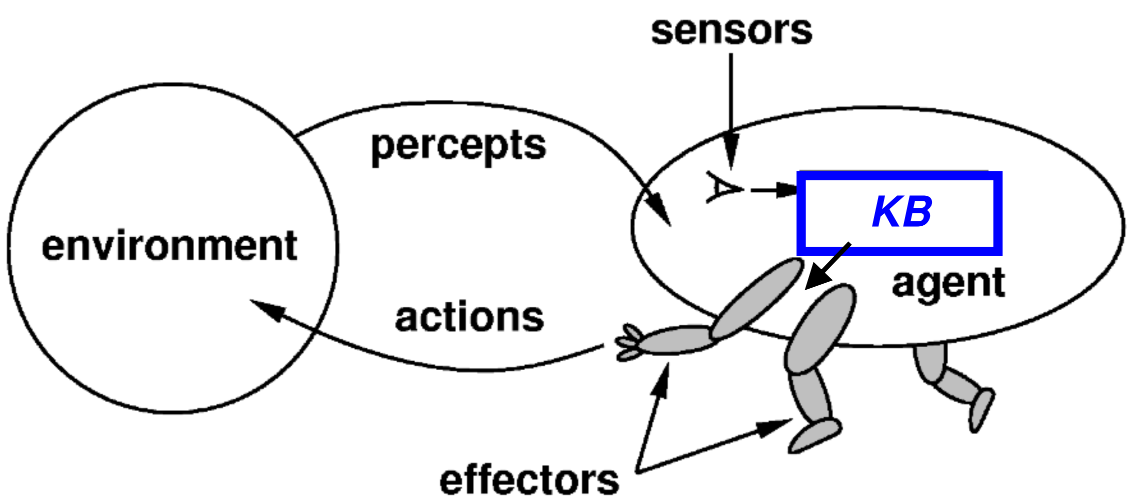
Volker Steinhage

Inhalt

- Rational denkende Agenten
- Die Wumpus-Welt
- Aussagenlogik: Syntax & Semantik
- Logische Folgerbarkeit
- Logische Ableitungen: Resolution

Rational denkende Agenten (1)

- Bisher lag das Schwergewicht auf *rational handelnden* Agenten



- Für komplexe Szenarien kann rationales Handeln auch *rationales Denken* bzw. *logisches Denken* durch den Agenten voraussetzen
- Für dieses rationale Denken sind relevante Aspekte der Welt symbolisch in einer *Wissensbasis* (*Knowledge base*, kurz: ***KB***) zur Verfügung zu stellen

Rational denkende Agenten (2)

Eine *Wissensbasis* (*Knowledge base*, kurz: ***KB***)

- enthält Sätze in einer sog. *Wissensrepräsentationssprache* (*Knowledge representation language*)

Syntax

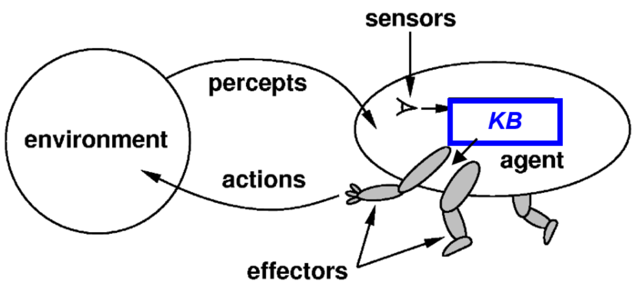
- mit einer *Wahrheitstheorie* (Logik), d.h. wir können Sätze als *Aussagen* über die Welt *interpretieren*

Semantik

Rational denkende Agenten (4)

Im wissensbasierten Agenten ist die Interaktion mit der KB vereinfacht durch *ASK* und *TELL*:

- *TELL(KB,a):* Ergänze KB um *a*.
- *ASK(KB,a):* Leite *a* aus KB ab.



function KB-Agent (*percept*) **returns** an *action*

static: *KB*, a knowledge base

t, a time counter (initially 0)

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*

Ein wissensbasierter Agent

Ein wissensbasierter Agent benutzt seine Wissensbasis also, um

(1) sein Hintergrundwissen zu repräsentieren,

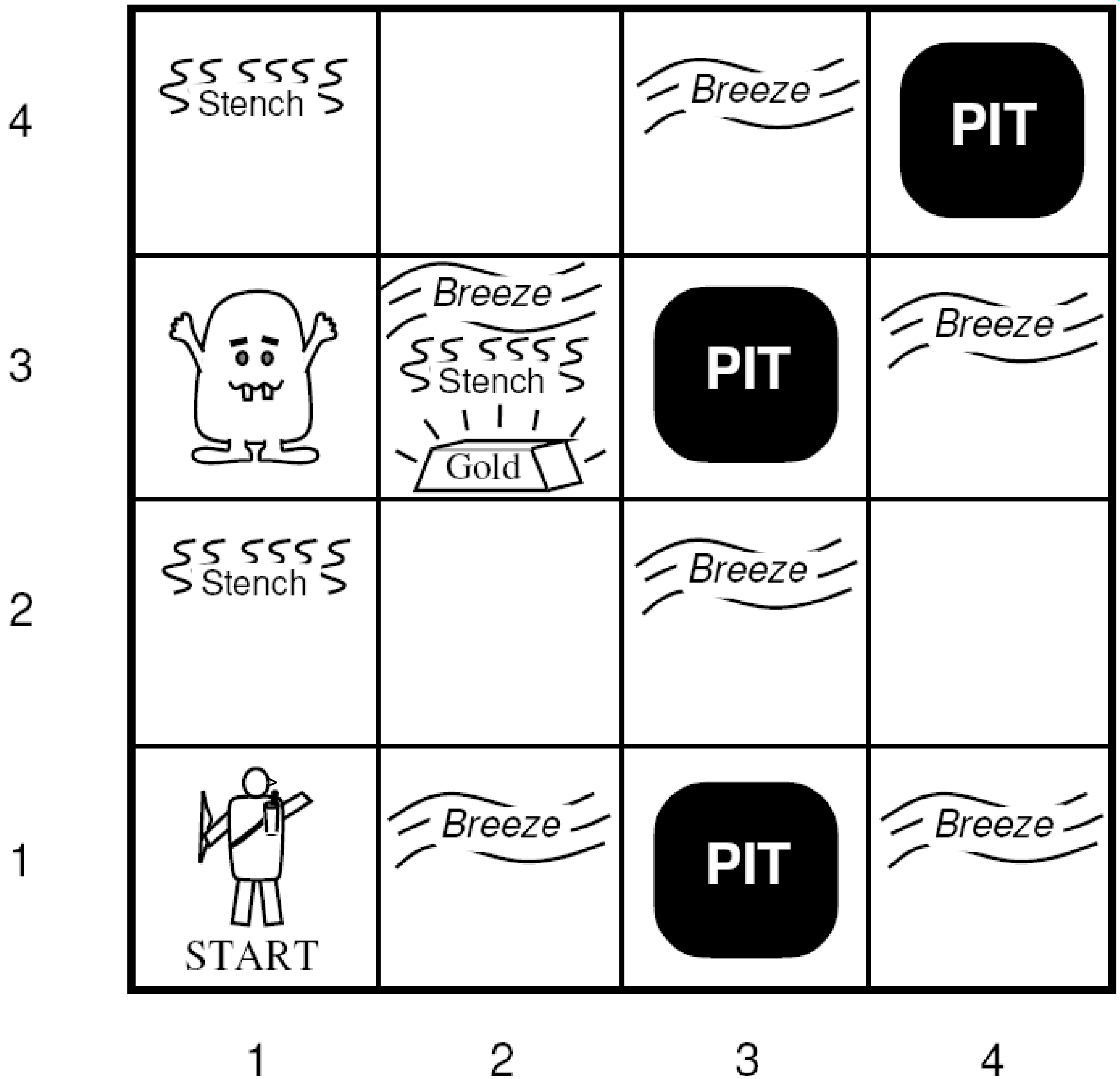
(2) seine Beobachtungen zu repräsentieren,

(3) daraus neue Aktionen abzuleiten,

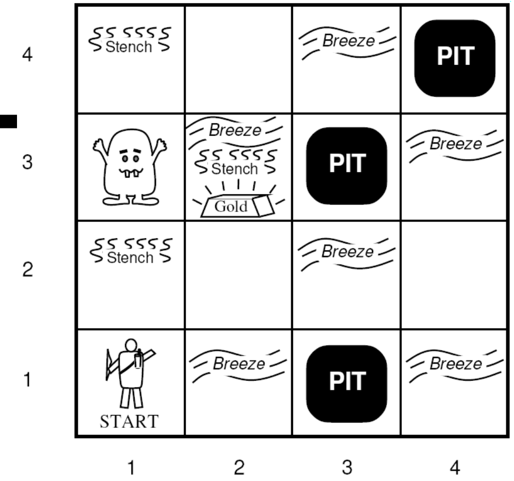
(4) die durchgeführten Aktionen zu repräsentieren.

```
function KB-Agent (percept) returns an action
    static: KB, a knowledge base (1)
            t, a time counter (initially 0)
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept,t)) (2)
    action ←ASK(KB,MAKE-ACTION-QUERY(t)) (3)
    TELL(KB, MAKE-ACTION-SENTENCE(action,t)) (4)
    t = t +1
    return action
```

Die Wumpus-Welt (1): Eine Beispielkonfiguration



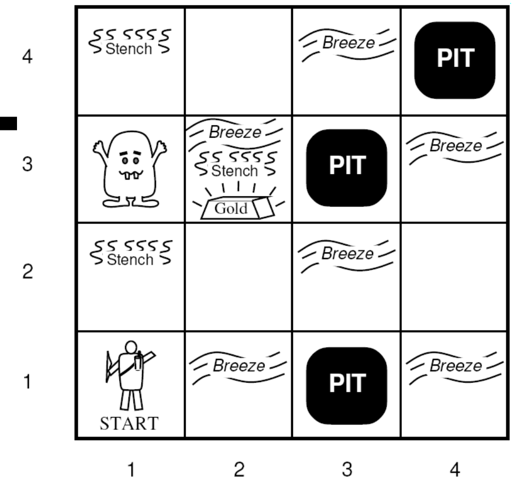
Die Wumpus-Welt (2)



Felder/Ereignisse und Perzepte

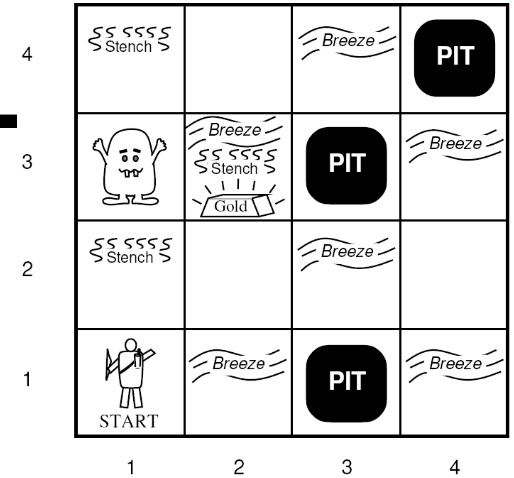
- Wumpus-Welt = eine 4×4-Umgebung mit 16 Feldern
- Im Feld des Wumpus‘ und in den direkt horizontal und vertikal benachbarten Feldern ist Geruch (Stench) wahrnehmbar
- Im Feld einer Fallgrube (Pit) und in den direkten horizontalen und vertikalen Nachbarfeldern ist Luftzug (Breeze) wahrnehmbar
- Im Feld mit dem Gold ist Glitzern (Glitter) wahrnehmbar
- Wenn der Agent in eine Wand läuft, bekommt er einen Stoß (Bump)
- Wird der Wumpus getötet, ist überall sein Schrei (Scream) wahrnehmbar
- Wahrnehmungen werden als binäre 5-Tupel dargestellt: z.B. bedeutet [Stench, Breeze, Glitter, -Bump, -Scream], dass Geruch, Zug und Glitzern, aber kein Stoß und kein Schrei wahrnehmbar sind

Die Wumpus-Welt (3)



- Aktionen und Zustände:
  - Aktionen:
    - greife ein Objekt im selben Feld
    - gehe vorwärts, 90° nach rechts drehen, 90° nach links drehen
    - schieße (es gibt nur einen Pfeil)
    - verlasse die Höhle (funktioniert nur im Feld [1,1])
  - Der Agent stirbt, wenn er in eine Fallgrube fällt oder dem lebenden Wumpus begegnet
  - Anfangszustand: Agent in [1,1] nach Osten orientiert, irgendwo 1 Wumpus, 1 Haufen Gold und 3 Fallgruben
  - *Ziel*: Hole das Gold und verlasse die Höhle $\rightsquigarrow$ 2 Teilziele

Die Wumpus-Welt (4)

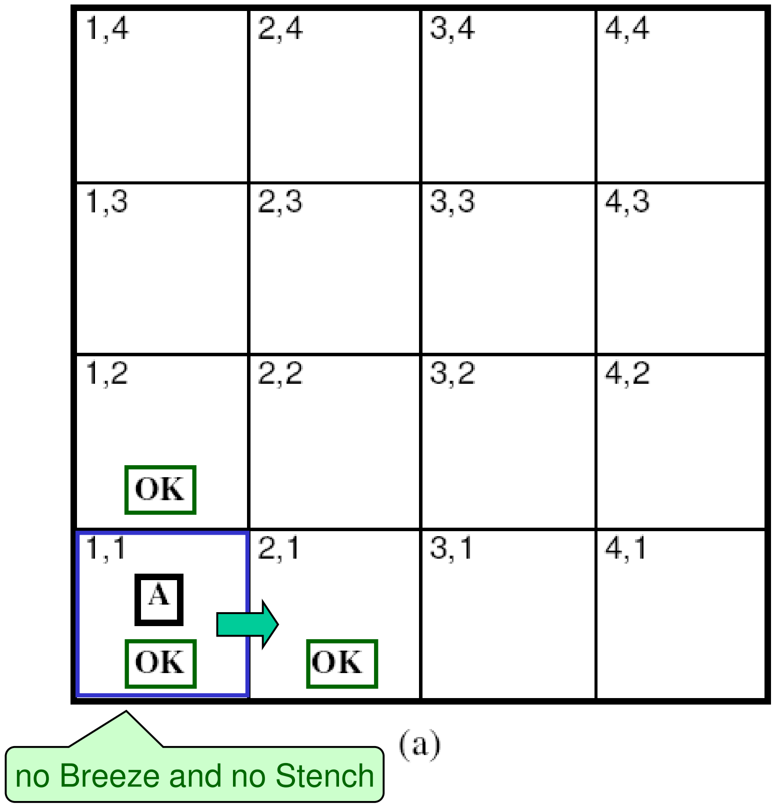


Wahrnehmungen, Inferenzen und Aktionen mit KB

(a) Agent in [1,1]: no Breeze, no Stench

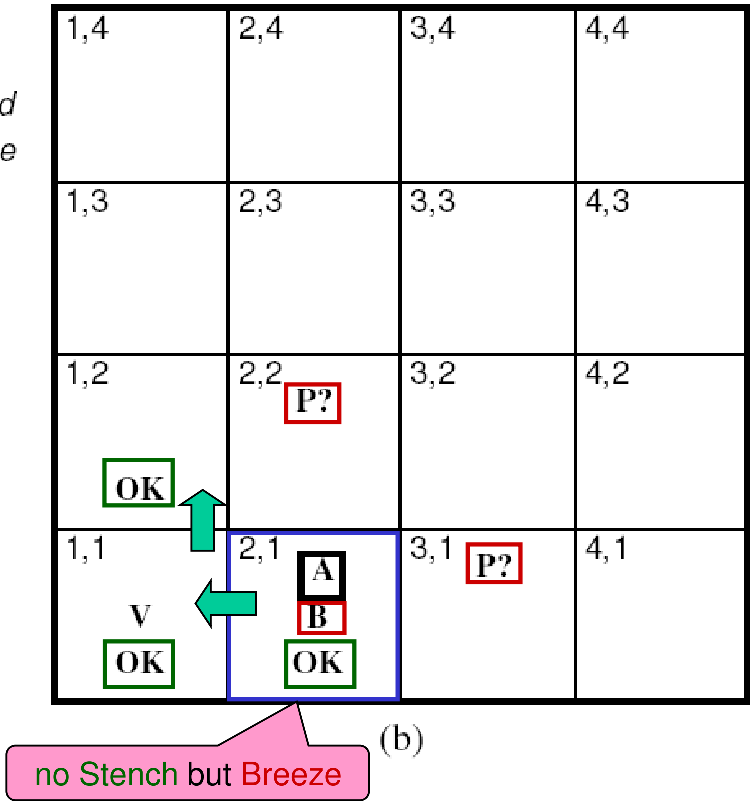
→ [1,2] und [2,1] sicher (OK) → gehe z.B. zu [2,1]

(b) Agent in [2,1]: Breeze → Pit in [2,2] oder [3,1] → gehe zu [1,2] über [1,1]



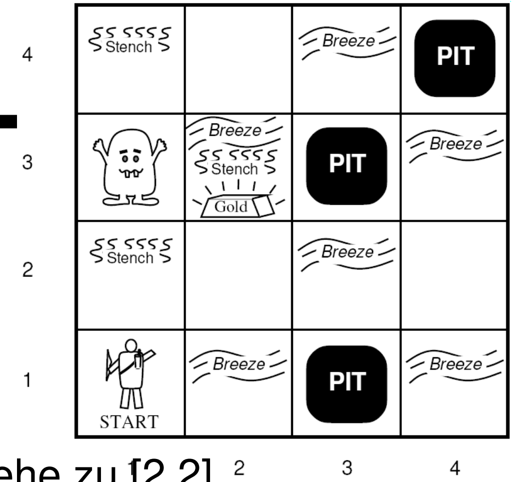
(a)

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



(b)

Die Wumpus-Welt (5)



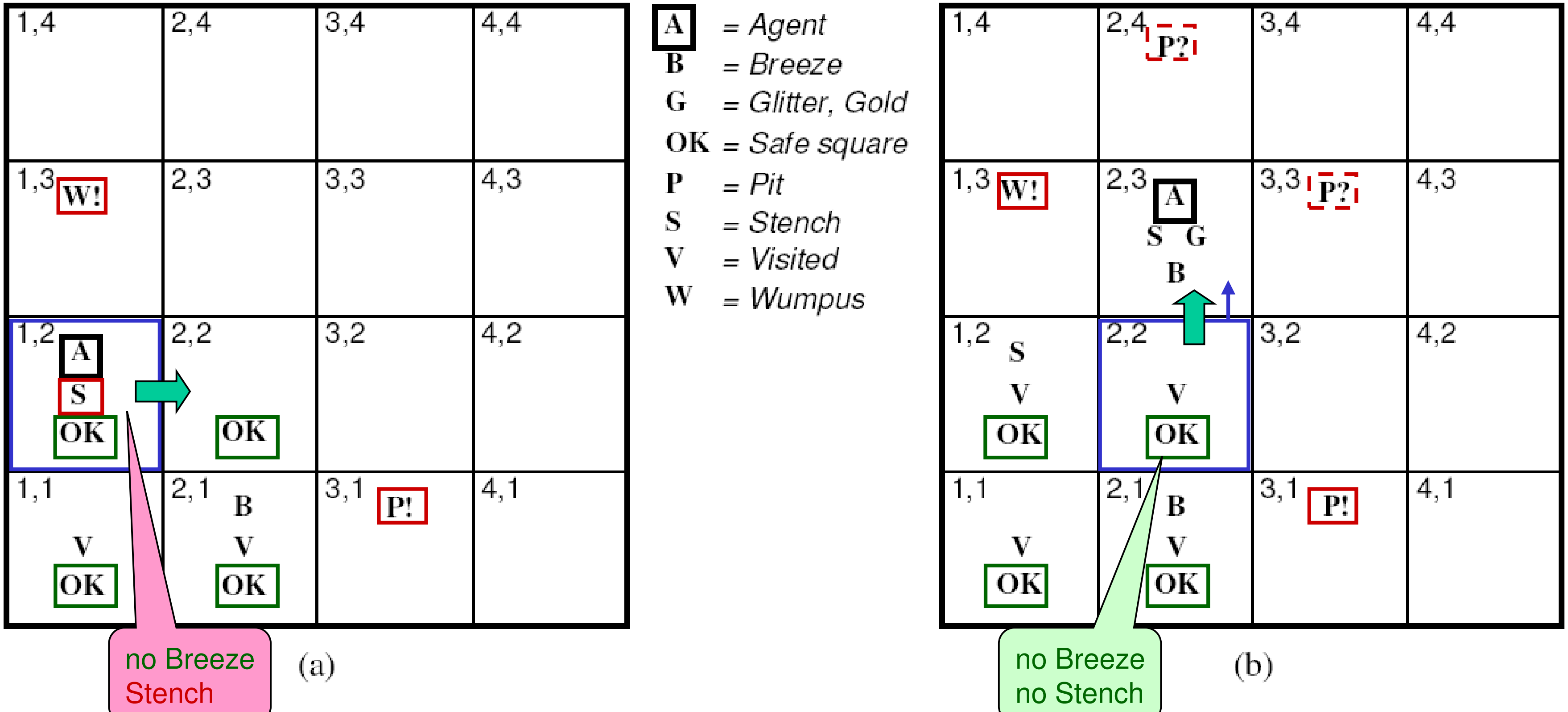
Wahrnehmungen, Inferenzen und Aktionen mit KB

(a) Agent in [1,2]: Stench in [1,2] → Wumpus in [1,3],

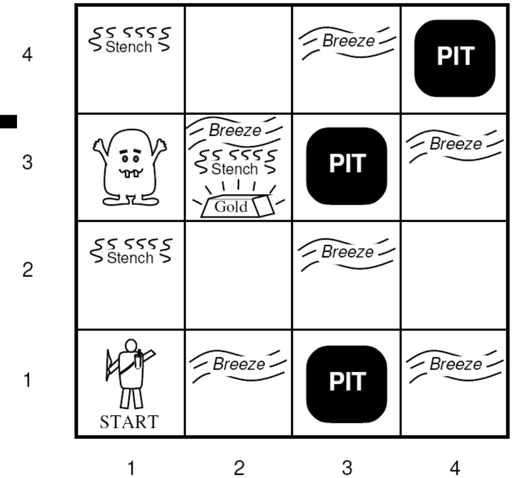
aber nicht in [2,2], da sonst Stench in [2,1] gewesen wäre KB

No Breeze in [1,2] → kein Pit in [2, 2], aber in [3,1] → [2,2] OK → gehe zu [2,2] KB

(b): No Breeze, no Stench, no Glitter in [2,2] → gehe z.B. zu [2,3] → 1. Erfolg: Gold ist gefunden

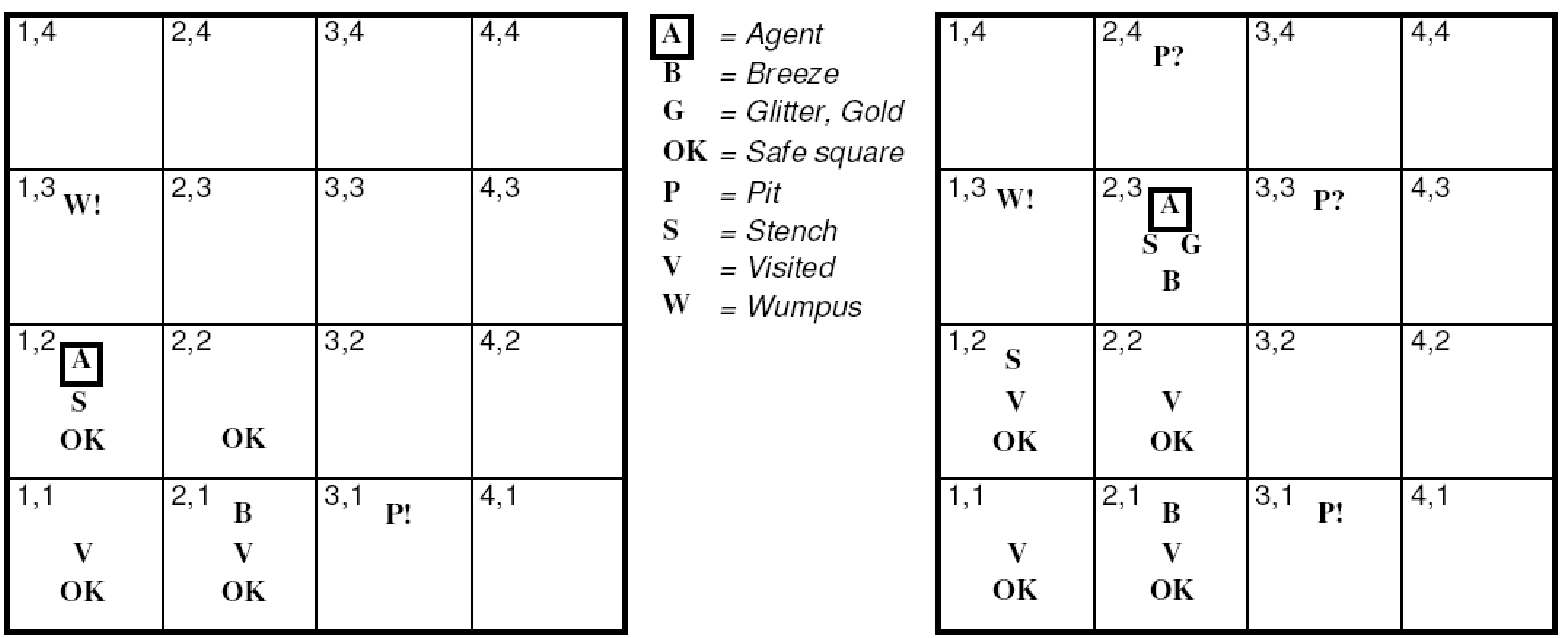


Die Wumpus-Welt (6)



Wie soll der Agent

- sein Wissen, z.B. „no Stench in [1,1]“,
- seine Vermutungen, z.B. „Pit in [3,1] oder [2,2]“,
- seine Schlüsse, z.B. „no Stench in [1,1] ⇒ no Wumpus in [1,2] und [2,1]“)
- seine bisherige Aktions- und Zustandsfolge repräsentieren und auf diesen arbeiten?



(a) (b)

Deklarative Sprachen

- Als *Wissensrepräsentationssprache* eignen sich insbes. deklarative Sprachen

- Einsatz einer *deklarativen Sprache* bedeutet:
  - in der *Wissensbasis* werden nur Fakten und Regeln der modellierten Welt repräsentiert
  - der prozedurale Aspekt – also das Arbeiten mit dem Wissen (hier das Ableiten von neuem Wissen) wird in einer anderen Komponente, nämlich der *Inferenzmaschine*, umgesetzt

➢ Vorteil der Übertragbarkeit zw. verschied. Domänen

➢ Eine *erste einfache Möglichkeit* dafür: die *Aussagenlogik*

Die Aussagenlogik kann „liefern“ (1)

Die Aussagenlogik (AL) kann Wissen repräsentieren:

- *Grundbausteine* der AL sind nicht weiter zerlegbare *atomare Aussagen*, von denen sinnvoll zu sagen ist, dass sie *wahr* oder *falsch* sind.*

  Beispiele:

  - „Der Block ist rot“
  - „Der Wumpus ist in [1,3]“
  - „Alan Turing war ein engl. Mathematiker, der die Turing-Maschine erfand.“

- Die AL hat *logische Konnektoren* wie „*und*“, „*oder*“, „*nicht*“, mit denen aus einfachen atomaren Aussagen komplexere *Formeln* konstruierbar sind.

---

* Dichotomie (Zweiteilung) der Aussagenlogik in wahre und falsche Aussagen.

Die Aussagenlogik „kann liefern“ (2)

Wir können in der Aussagenlogik schlussfolgern!

- Die AL lässt die Beantwortung zu, ob ist eine Aussage *wahr* ist.
- Die AL lässt die Beantwortung zu, ob eine Aussage aus einer Wissensbasis KB *folgt* (i.Z.: KB $\models \varphi$).

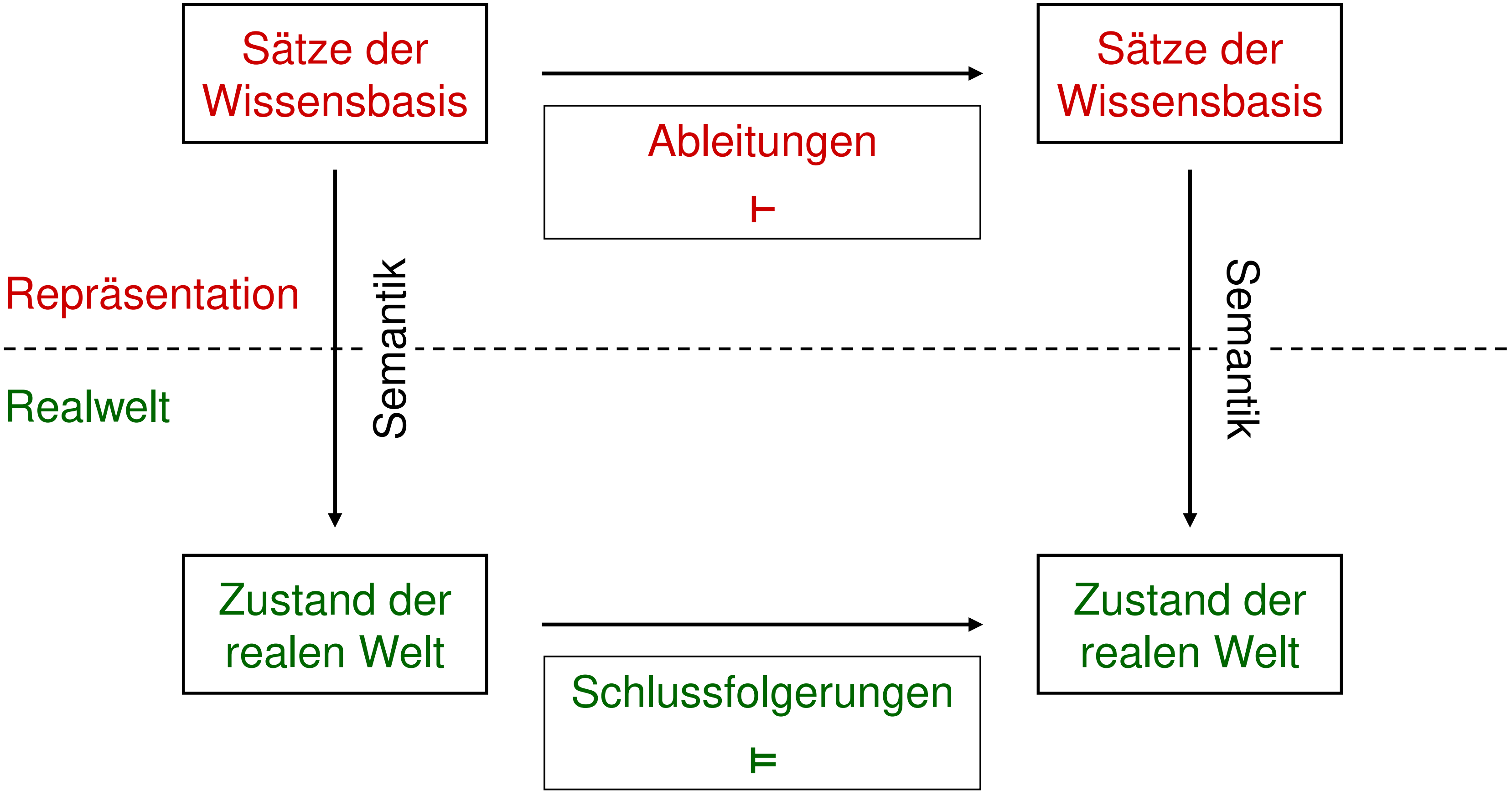
Semantik

- Für die AL gibt es einen syntaktischen *Ableitungsbegriff* (i.Z.: KB $\vdash \varphi$), der mit dem *Folgerungsbegriff* korrespondiert.

Syntax

Die Aussagenlogik „kann liefern“ (3)

Schlussfolgern mit der AL:



Die Aussagenlogik „kann liefern“ (4)

- Wir können mit der Aussagenlogik die Bedeutung und Implementierung von ASK und TELL festschreiben und umsetzen!

function KB-Agent (*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a time counter (initially 0)
  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*

Zur Erinnerung: Syntax der AL (1)

Geg.: (1) abzählbares Alphabet Σ von *atomaren Aussagen* bzw. *atomaren Sätzen*:
$True$, $False$, P, Q, R, . . . ,

(2) *logische Verknüpfungen*: Negation ($\neg$), Konjunktion ($\wedge$), Disjunktion ($\vee$), Implikation ($\Rightarrow$) und Äquivalenz ($\Leftrightarrow$).

Daraus ergeben sich alle *aussagenlogische Formeln* bzw. *Sätze:*

Satz	$\rightarrow$ atomarerSatz | komplexerSatz
atomarerSatz	$\rightarrow$ $True$ | $False$ | Symbol
Symbol	$\rightarrow$ P | Q | R | …
komplexerSatz	$\rightarrow$ $\neg$ Satz | Satz $\wedge$ Satz | Satz $\vee$ Satz
	| Satz $\Rightarrow$ Satz | Satz $\Leftrightarrow$ Satz

Operator-Präzedenz zur Vermeidung von zu vielen Klammerungen: $\neg$ vor $\wedge$ vor $\vee$ vor $\Rightarrow$ vor $\Leftrightarrow$

Zur Erinnerung: Syntax der AL (2)

Weitere wichtige Terminologie für das Folgende:

Atom: atomarer Satz, z.B. P, Q, $True$

Literal: positives oder negiertes Atom, z.B. *(+)*P, $\neg P$ *(oder* $\overline{P}$ *oder* -P*)*

Klausel: Disjunktion bzw. Konjunktion von Literalen,

z.B. $P \vee \neg Q \vee R$ bzw. $Q \wedge R \wedge \neg S$

Semantik (Kurzform)

Eine *Interpretation* über den Atomen von Σ ist eine Funktion $\mathcal{I} : \Sigma \to \{$ *True, False* $\}$:

$\mathcal{I} \models$ *True* (*True* ist unter jeder Interpretation wahr)

$\mathcal{I} \not\models$ *False* (*False* ist unter jeder Interpretation falsch)

$\mathcal{I} \models P$ gdw. $\mathcal{I}(P) =$ *True*

$\mathcal{I} \models \varphi$ in Worten:

„$\mathcal{I}$ *erfüllt* φ" oder „φ ist wahr unter $\mathcal{I}$"

Die Interpretation $\mathcal{I}(\varphi)$ einer Formel φ:*

$\mathcal{I} \models \neg\varphi$ gdw. $\mathcal{I} \not\models \varphi$

$\mathcal{I} \models \varphi \wedge \psi$ gdw. $\mathcal{I} \models \varphi$ und $\mathcal{I} \models \psi$ für Formeln φ, ψ

$\mathcal{I} \models \varphi \vee \psi$ gdw. $\mathcal{I} \models \varphi$ oder $\mathcal{I} \models \psi$ für Formeln φ, ψ

$\mathcal{I} \models \varphi \Rightarrow \psi$ gdw. wenn $\mathcal{I} \models \varphi$, dann $\mathcal{I} \models \psi$ für Formeln φ, ψ

$\mathcal{I} \models \varphi \Leftrightarrow \psi$ gdw. $\mathcal{I} \models \varphi$ genau dann, wenn $\mathcal{I} \models \psi$ für Formeln φ, ψ

* Statt $\mathcal{I}(\varphi)$ kann auch geschrieben werden $\varphi^{\mathcal{I}}$

Beispiel

Geg.: Interpretation I mit

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

Geg.: Formel φ mit

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

Frage: $I \models \varphi$?

Übung

Modelle und Entscheidbarkeit

Eine Interpretation $\mathcal{I}$ heißt *Modell* der Formel φ,

wenn $\mathcal{I}$ die Formel φ erfüllt (i.Z.: $\mathcal{I} \vDash \varphi$).

Eine Interpretation $\mathcal{I}$ heißt *Modell* einer *Menge von Formeln*,

wenn $\mathcal{I}$ Modell aller Formeln der Menge ist.

Eine Formel φ heißt

- *erfüllbar*, wenn es mind. ein Modell $\mathcal{I}$ für φ gibt: $\exists \mathcal{I} : \mathcal{I} \vDash \varphi$.
- *unerfüllbar* (*inkonsistent*), wenn es kein Modell $\mathcal{I}$ für φ gibt: $\forall \mathcal{I} : \mathcal{I} \nvDash \varphi$.
- *falsifizierbar*, wenn es mind. eine nicht erfüllende Interpr. $\mathcal{I}$ für φ gibt: $\exists \mathcal{I} : \mathcal{I} \nvDash \varphi$.
- *allgemeingültig* (*tautologisch*), wenn jede Interpr. $\mathcal{I}$ Modell von φ ist: $\forall \mathcal{I} : \mathcal{I} \vDash \varphi$.

Wie entscheiden wir, ob eine Formel erfüllbar, allgemeingültig usw. ist?

Die Wahrheitstabellenmethode

- Die einfachste Methode zur Überprüfung einer Formel auf Erfüllbarkeit, Allgemeingültigkeit usw. ist die Aufstellung der Wahrheitstabelle!

- Beispiel: Ist $\varphi = ((P \vee H) \wedge \neg H) \Rightarrow P$ allgemeingültig?

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

$\rightsquigarrow$ φ ist unter allen Wahrheitsbelegungen wahr $\rightsquigarrow$ φ ist allgemeingültig.

Wertetabellen vs. Inferenz

Die Wertetabellenmethode hat leider exponentielle Komplexität:

- Die Wertetabellenmethode ist eine *exhaustive Aufzählung* aller Interpretation der Formel φ

⇝ Wenn Formel φ insgesamt N verschiede Symbole enthält, dann enthält die Wertetabelle insgesamt 2^N Einträge (Zeilen)

Inferenz stellt die effizientere Variante gegenüber der Wertetabellen-methode dar … und *Normalformen* steigern zusätzlich die Effizienz der Inferenz!

⇝ Weiter also mit wichtigen *Normalformen* und relevantem *Inferenzkalkül*

Normalformen

Eine Formel φ ist in *konjunktiver Normalform* (*KNF*), wenn φ eine Konjunktion von disjunktiven Klauseln (Disjunktionen von Literalen $l_{i,j}$) ist:

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

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

Eine Formel φ ist in *disjunktiver Normalform* (*DNF*), wenn φ eine Disjunktion von konjunktiven Klauseln (Konjunktionen von Literalen $l_{i,j}$) ist:

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

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

Es gilt: Zu *jeder* Formel φ existieren *äquivalente Formeln in KNF und DNF*

Normalformen → Klauselmengen (1)

- Formeln in KNF oder DNF lassen sich als Klauselmengen darstellen!

Dabei verzichtet man auf das Schreiben von „$\vee$“ und „$\wedge$“.

$(\neg P \vee Q)$
$\wedge \quad (\neg Q \vee R)$
$\rightsquigarrow \quad \{\{\neg P, Q\}, \{\neg Q, R\}\}$

- Für eine Formel φ in KNF: $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m_i} l_{i,j}).$
  - Jede Disjunktion von Literalen wird als Literalmenge $\{l_{i,j}\}$ ($j \in \{1,\ldots,m_i\}$) notiert und als Klausel* C_i bezeichnet
  - Die Zahl der Literale eine Klausel wird als Länge der Klausel bezeichnet
  - Die Konjunktion aller Klauseln C_i wird als Klauselmenge $\Delta = \{C_i\}$ ($i \in \{1,\ldots,n\}$) notiert
  - Die *leere Klausel* wird mit $\square$ notiert und ist unerfüllbar

* Engl.: clauses – im Deutschen auch: Clausen, Klausen.

Normalformen $\rightarrow$ Klauselmengen (2)

- Zum einfachen Lesen werden Klauselmengen häufig als „Matrizen“ geschrieben: Klauseln bilden die Zeilen, in deren Spalten die Literale stehen

- Für eine Wissensbasis KB erlaubt die KNF eine einfache Schreib- und Leseweise von Fakten und Regeln:

$Q \Rightarrow P \rightsquigarrow \neg Q \vee P$

KB:

$$\begin{array}{lll} A & & \\ B & & \\ \overline{A} & \overline{B} & C \end{array}$$

$A \wedge B \Rightarrow C$

- Daher wird im Weiteren nur die KNF weiter verfolgt

Erzeugen der KNF

Vier Schritte:

- Eliminiere „$\Rightarrow$“ und „$\Leftrightarrow$“ : $\alpha \Rightarrow \beta \rightarrow (\neg \alpha \vee \beta)$ usw.
- Schiebe „$\neg$“ nach innen: $\neg(\alpha \vee \beta) \rightarrow (\neg \alpha \wedge \neg \beta)$ usw.
- Verteile: „$\vee$“ über „$\wedge$“ : $((\alpha \wedge \beta) \vee \gamma) \rightarrow ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$ usw.
- Vereinfache: $(\alpha \vee \alpha) \rightarrow \alpha$ usw.

Übung

Ergebnis ist eine Konjunktion von Disjunktionen von Literalen!

$\rightsquigarrow$ Weiter mit *Schlussfolgern* und relevantem *Inferenzkalkül*

Logische Folgerbarkeit

Formel φ folgt aus KB (i.Z.: $KB \models \varphi$), wenn φ in allen Modellen von KB wahr ist:

$KB \models \varphi$ gdw. $\mathcal{I} \models \varphi$ für alle Modelle $\mathcal{I}$ von KB.

Einige beweisbare Eigenschaften der Folgerungsbeziehung:

- Deduktionssatz: $KB \cup \{\varphi\} \models \psi$ gdw. KB $\models \varphi \Rightarrow \psi$
- Kontrapositionssatz: $KB \cup \{\varphi\} \models \neg\psi$ gdw. KB $\cup \{\psi\} \models \neg\varphi$
- Widerspruchssatz: $KB \cup \{\varphi\}$ ist unerfüllbar gdw. $KB \models \neg\varphi$

Frage:

- Können wir $KB \models \varphi$ entscheiden, ohne alle Interpretationen betrachten zu müssen (Wahrheitstabellenmethode)?

Inferenzen, Kalküle und Beweise

Kernidee und Terminologie*:*

- *Kalkül:* Menge von *Inferenzregeln* und logischen Axiomen
- *Beweisschritt:* Anwendung einer *Inferenzregel* auf eine Menge von Formeln
- *Beweis:* Sequenz von Beweisschritten, wobei
  - die mit jedem Schritt neu abgeleiteten Formeln zu *KB* hinzugefügt werden
  - im letzten Schritt die *Zielformel* erzeugt wird

Korrektheit und Vollständigkeit

Falls es bei Wissensbasis *KB* in einem Kalkül *C* einen Beweis für eine Formel φ gibt, schreiben wir:*

$$KB \vdash_C \varphi$$

Ein Kalkül C heißt *korrekt*, wenn alle aus einer *KB* ableitbaren Formeln auch tatsächlich logisch folgen:

$$KB \vdash_C \varphi \text{ impliziert } KB \models \varphi$$

(folgt i.A. aus Korrektheit der Inferenzregeln und der logischen Axiome)

Ein Kalkül heißt *vollständig*, falls jede Formel, die logisch aus KB folgt, auch aus *KB* ableitbar ist:

$$KB \models \varphi \text{ impliziert } KB \vdash_C \varphi$$

* ohne Subskript *C*, wenn *C* eindeutig

Das Resolutionskalkül: Repräsentation

Voraussetzung: Alle Formeln in *KB* liegen als Klauseln in KNF vor

Es bezeichnen

- Δ eine Menge von Klauseln
- C eine Klausel und $\square$ die leere Klausel
- l ein Literal und $\bar{l}$ bzw. $\neg l$ dessen Negation

Dann gilt:

- Eine Interpretation $\mathcal{I}$ erfüllt C gdw. es ein $l \in C$ gibt, so dass $\mathcal{I} \models l$
- Keine Interpretation $\mathcal{I}$ erfüllt $\square$
- $\mathcal{I}$ erfüllt Δ, falls für alle $C \in \Delta : \mathcal{I} \models C$
- $\mathcal{I} \not\models \square$, $\mathcal{I} \not\models \{\square\}$, $\mathcal{I} \models \{\ \}$ für alle $\mathcal{I}$

Das Resolutionskalkül: die Resolutionsregel

Die Regel der (*Grund-)Resolution:**

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

Mengenschreibweise von Klauseln

- $C_1 \cup C_2$ wird (*Grund-)Resolvente*

  der *Elternklauseln* $C_1 \cup \{\, l \,\}$ und $C_2 \cup \{\, \bar{l} \,\}$ genannt

- l und $\bar{l}$ sind die *Resolutionsliterale*

Beispiel: $\{a, b, \neg c\}$ und $\{a, d, c\}$ sind

über c und $\neg c$ resolvierbar zu $\{a, b, d\}$.

---

* Wir sprechen von Grundresolution, solange wir die Resolution in der Aussagenlogik anwenden. Später mehr dazu.

Das Resolutionskalkül: Resolutionsableitungen

Die Anwendung der Grundresolution auf eine Klauselmenge Δ:

$$R(\Delta) = \Delta \cup \{C \mid C \text{ ist Resolvente zweier Klauseln aus } \Delta\}$$

Wir sagen, dass Klausel D aus Δ mit Hilfe von (Grund-)Resolution *abgeleitet* werden kann, i.Z.

$$\Delta \vdash D,$$

wenn es $C_1, C_2, \ldots, C_n = D$ gibt, so dass

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

Korrektheit der (Grund)-Resolution

Lemma: (*Korrektheit der (Grund-)Resolution*):

Seien $C_1 = C_1' \vee l$

und $C_2 = C_2' \vee \neg l$,

dann ist die Resolvente $C_1' \vee C_2'$ logische Folgerung von C_1 *und* C_2

Beweisidee: Fallunterscheidung über l oder Wahrheitswerttabelle

Satz: (*Korrektheit der Ableitung durch (Grund-)Resolution*):

Wenn $\Delta \vdash D$, dann $\Delta \models D$

Beweisidee: Da alle $D \in R(\Delta)$ aus Δ logisch folgen, ergibt sich der Satz durch Induktion über die Länge der Ableitung

Vollständigkeit der (Grund)-Resolution

Ist die (Grund-)Resolution auch vollständig:

$\Delta \vDash \varphi$ *impliziert* $\Delta \vdash \varphi$?

…. zunächst höchstens für Klauseln!

… aber: $\left\{ \{a, b\}, \{\neg b, c\} \right\} \begin{array}{ll} \vDash & \{a, b, c\} \\ \nvdash & \{a, b, c\} \end{array}$

Widerspruchsvollständigkeit der (Grund)-Resolution

Jedoch ist die (Grund-)Resolution widerlegungsvollständig!

Definition: Eine Ableitung von $\square$ aus Δ heißt *Widerlegung* von Δ

(Grund-)Resolutions-Theorem: *

$$\Delta \quad \text{ist inkonsistent gdw. } \Delta \vdash \square$$

D.h. wir können $KB \vDash \varphi$ zeigen mit Hilfe des Widerspruchssatzes (s. Folie 29):

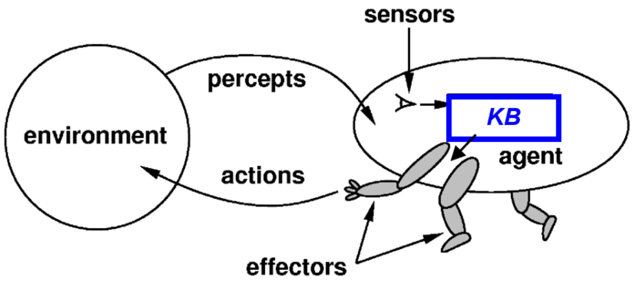
$KB \cup \{\neg\varphi\}$ *ist inkonsistent* gdw. $KB \vDash \varphi$

* Anschaulicher Beweis über semantische Bäume im Anhang.

Vollständigkeit der (Grund)-Resolution

Fazit:

- Die Aussagenlogik mit der Grundresolution scheint geeignet zu sein für den Aufbau der Wissensbasis eines wissensbasierten Agenten, d.h. um
  - sein Hintergrundwissen zu repräsentieren
  - seine Beobachtungen zu speichern
  - daraus neue Aktionen abzuleiten
  - die durchgeführten Aktionen zu speichern



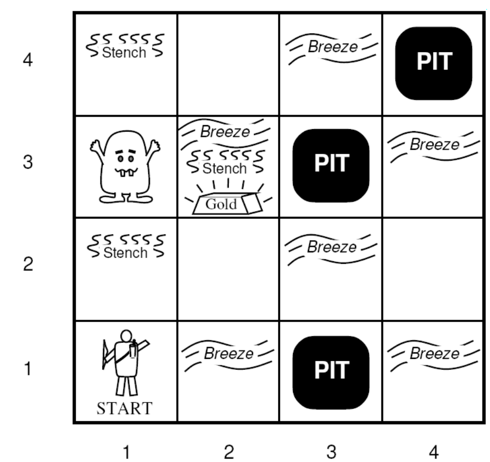
- Aussagenlogik und Grundresolution können
  - einfache und komplexe Aussagen repräsentieren
  - korrekte und widerlegungsvollständige Ableitungen umsetzen

Die Grundresolution für die Wumpus-Welt? – Die Situation

1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 $\boxed{\mathbf{A}}$ **S** **OK**	2,2 **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

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

Historie: Zuvor von [1,1] erst nach [2,1], dann über [1,1] nach [1,2].



Grundresolution für die Wumpus-Welt: Situationswissen

- (Fakten-)Wissen über die Situation (u.a.) $\rightarrow$ Fakten-Klauseln $F_1, F_2, \ldots$

$F_1: \neg S_{1,1}$ $\quad$ $F_2: \neg B_{1,1}$

$F_3: \neg S_{2,1}$ $\quad$ $F_4: \ B_{2,1}$

$F_5: \ S_{1,2}$ $\quad$ $F_6: \neg B_{1,2}$

mit $B_{i,j}$ = Breeze in (i, j), $S_{i,j}$ = Stench in (i, j)

- (Regel-)Wissen über die Wumpus-Welt: $\rightarrow$ Regel-Klauseln $R_1, R_2, \ldots$

$R_1: \quad \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

$R_2: \quad \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$

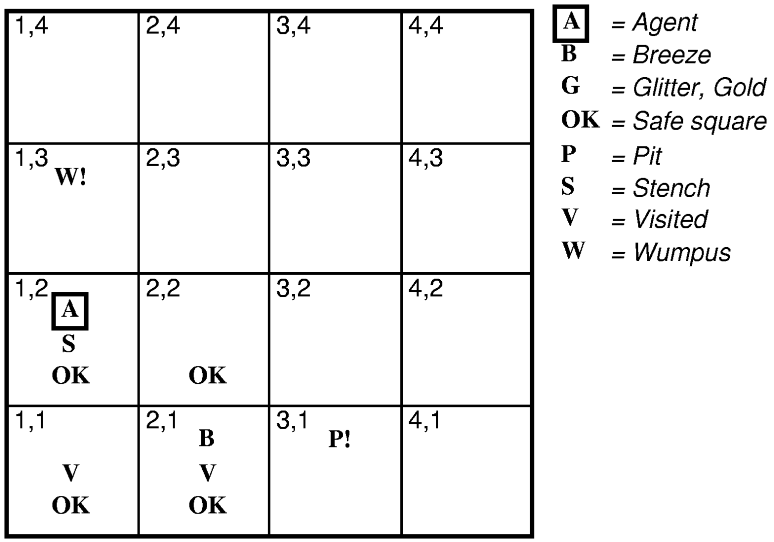
$R_3: \quad \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$

$R_4: \quad S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

...

- Ziel: zeige, dass der Wumpus in Feld (1,3) ist

$KB \models W_{1,3}$



Grundresolution für die Wumpus-Welt: KNF

- Situationswissen als Faktenklauseln der Länge 1 (sog. 1-Klauseln):

F_1: $\neg S_{1,1}$, F_2: $\neg B_{1,1}$, F_3: $\neg S_{2,1}$, F_4: $B_{2,1}$, F_5: $S_{1,2}$, F_6: $\neg B_{1,2}$, . . .

- Regelwissen als Regelklauseln der Länge n > 1:

$R_{1,1}$: $S_{1,1}$, $\neg W_{1,1}$, $R_{1,2}$: $S_{1,1}$, $\neg W_{1,2}$, $R_{1,3}$: $S_{1,1}$, $\neg W_{2,1}$, (drei 2-Klauseln)

$R_{2,1}$: $S_{2,1}$, $\neg W_{1,1}$, $R_{2,2}$: $S_{2,1}$, $\neg W_{2,1}$, $R_{2,3}$: $S_{2,1}$, $\neg W_{2,2}$, $R_{2,4}$: $S_{2,1}$, $\neg W_{3,1}$, (vier 2-Klauseln)

$R_{3,1}$: $S_{1,2}$, $\neg W_{1,1}$, $R_{3,2}$: $S_{1,2}$, $\neg W_{1,2}$, $R_{3,3}$: $S_{1,2}$, $\neg W_{2,2}$, $R_{3,4}$: $S_{1,2}$, $\neg W_{1,1}$, (vier 2-Klauseln)

R_4 : $\neg S_{1,2}$, $W_{1,3}$, $W_{1,2}$, $W_{2,2}$, $W_{1,1}$ (eine 5-Klausel)

Übung

. . .

- Negierte Zielformel für widerspruchsvollständige Grundresolution:

Z: $\neg W_{1,3}$

Resolutionsbeweis für die Wumpus-Welt

Ein Beweis durch Resolution über acht abgeleitete Klausen A_i $(i \in \{1,..,8\})$:

- $Z: \neg W_{1,3}$ — $R_4: \neg S_{1,2} \lor W_{1,3} \lor W_{1,2} \lor W_{2,2} \lor W_{1,1}$
  - A_1 aus $R(Z,R_4)$: $\neg S_{1,2} \lor W_{1,2} \lor W_{2,2} \lor W_{1,1}$
- $F_5: S_{1,2}$ — $A_1: \neg S_{1,2} \lor W_{1,2} \lor W_{2,2} \lor W_{1,1}$
  - A_2 aus $R(F_5,A_1)$: $W_{1,2} \lor W_{2,2} \lor W_{1,1}$
- $F_1: \neg S_{1,1}$ — $R_{1,1}: S_{1,1} \lor \neg W_{1,1}$
  - A_3 aus $R(F_1,R_{1,1})$: $\neg W_{1,1}$
- $A_3: \neg W_{1,1}$ — $A_2: W_{1,2} \lor W_{2,2} \lor W_{1,1}$
  - A_4 aus $R(A_3,A_2)$: $W_{1,2} \lor W_{2,2}$
- $F_3: \neg S_{2,1}$ — $R_{2,3}: S_{2,1} \lor \neg W_{2,2}$
  - A_5 aus $R(F_3,R_{2,3})$: $\neg W_{2,2}$
- $A_5: \neg W_{2,2}$ — $A_4: W_{1,2} \lor W_{2,2}$
  - A_6 aus $R(A_5,A_4)$: $W_{1,2}$
- $F_1: \neg S_{1,1}$ — $R_{1,2}: S_{1,1} \lor \neg W_{1,2}$
  - A_7 aus $R(F_1,R_{1,2})$: $\neg W_{1,2}$
- $A_6: W_{1,2}$ — $A_7: \neg W_{1,2}$
  - $A_8 = \square$ aus $R(A_6,A_7)$

Übung

Von Wissen zu Aktionen

Wir können jetzt neue Fakten inferieren,

aber wie setzen wir das Wissen in Aktionen um?

1) Negative Selektion: Schließe alle beweisbar gefährlichen Aktionen aus

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

2) Positive Selektion: Schlage nur Aktionen vor, die beweisbar sicher sind

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

Aus den Vorschlägen muss der Agent sich noch eine Aktion „aussuchen"!

---

* $A_{1,1}$ = Agent in Feld (1,1), $East_A$ = Agent nach Osten orientiert (im folg. Algm. als „*right*" bezeichnet), $W_{2,1}$ = Wumpus in Feld (2,1),

Der aussagenlogische Wumpus–Agent (1)

function PL-Wumpus-Agent (percept) **returns** an action

PL hier für propositional logic

inputs: percept, a list of [*stench, breeze, glitter, bump, scream*]

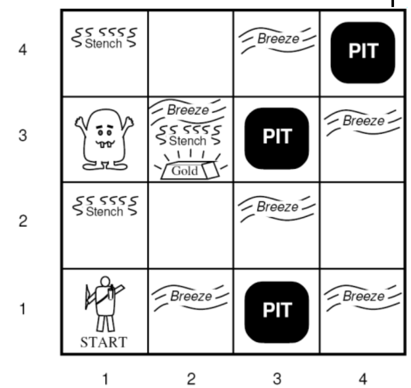
static: *KB*, a knowledge base, initially containing the “physics” of the wumpus world,

x, y, orientation, the agent’s position (initially [1,1]) and orientation (initially *right*),

visited, an array indicating which squares have been visited, initially *false* for all squares,

action, the agent’s most recent action, initially *null*,

plan, an action sequence, initially *empty*



update *x, y, orientation*, *visited* based on *action*

if *stench* **then** TELL($KB, S_{x,y}$) **else** TELL($KB, \neg S_{x,y}$)

if *breeze* **then** TELL($KB, B_{x,y}$) **else** TELL($KB, \neg B_{x,y}$)

Aktualisierungen durch Aktionsausführung und Sensorik

if *glitter* **then** *action* ← *grab*

else if *plan* is *nonempty* **then** *action* ← POP(*plan*)

fringe square = unbesuchtes Feld benachbart zu besuchtem Feld

else if for some fringe square [i,j], ASK($KB, \neg P_{i,j} \wedge \neg W_{i,j}$) is entailed by KB **or** // provably save

for some fringe square [i,j], ASK($KB, P_{i,j} \vee W_{i,j}$) is not entailed by KB **then do** // poss. save

plan ← A*-GRAPH-SEARCH(ROUTE-PROBLEM([*x,y*], *orientation*, [*i,j*], *visited*)) *

action ← POP(*plan*)

else *action* ← a randomly chosen move

return *action*

- ROUTE-PROBLEM erzeugt ein Suchproblem, dessen Lösung eine Aktionsfolge ist, die von [x,y] zu [i,j] führt und nur über besuchte Felder verläuft.

Der aussagenlogische Wumpus–Agent (2)

Das aussagenlogische Wumpus–Agentenprogramm PL-Wumpus-Agent

- aktualisiert seine Wissensbasis über TELL,
  - ob im aktuell besuchten Feld Geruch oder Luftzug wahrnehmbar sind
  - welche Felder schon besucht wurden
- wählt die nächste Aktion aus:
  - greift das Gold, wenn dieses im aktuellen Feld ist,
  - führt sonst den nächsten Schritt des Planes durch,
  - sucht sonst ein nächstes unbesuchtes Feld (*fringe square*) und generiert einen Plan, um vom aktuellen Feld zu diesem unbesuchten Feld zu kommen
  - führt sonst eine zufällige Bewegungsaktion aus.

Probleme mit der Aussagenlogik (1)

Obwohl Aussagenlogik für die Darstellung der WUMPUS-Welt ausreichend scheint, ist sie doch ziemlich umständlich.

1. *Allgemeine Regeln* müssen für *jedes einzelne Feld* aufgestellt werden ↯

$$R_1 : \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$$

$$R_2 : \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$$

$$R_3 : \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$$

$$\ldots$$

2. Tatsächlich müssten wir i. A. alle atomaren Aussagen *mit einem Zeitindex* versehen, der die zeitliche Gültigkeit der Aussage in der Zeit beschreibt $\leadsto$ weitere Aufblähung der Regelmenge. ↯

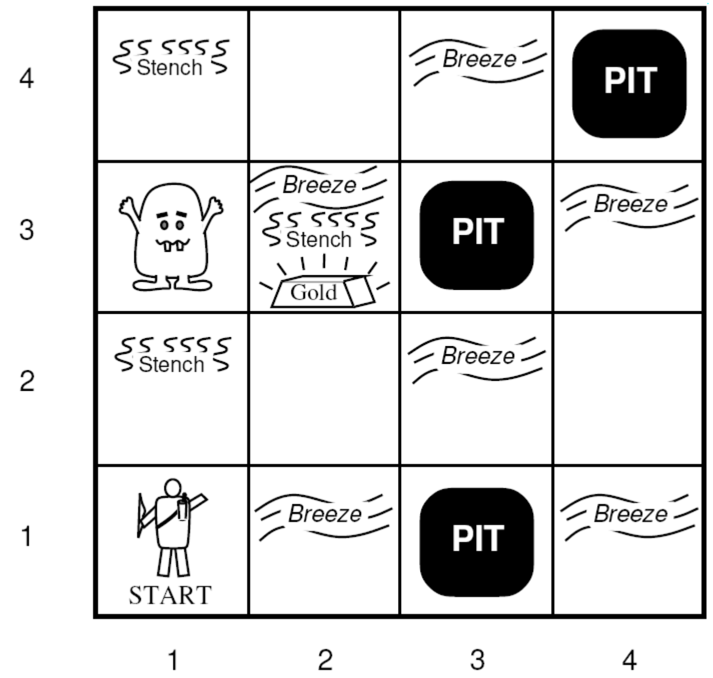
Probleme mit der Aussagenlogik (2)

- Letztlich gibt uns die Aussagenlogik keine sprachlichen Mittel an die Hand, mit denen die Zusammenhänge der repräsentierten Welt differenzierter ausdrücken können. ↯

⇝ Eine *mächtigere Logik wäre wünschenswert*, in der wir *Objekte* und *Objekt-variable* sowie deren *Eigenschaften* und *Relationen* beschreiben können!

⇝ Ein Ausweg könnte die nächst höhere Logik sein, die *Prädikatenlogik 1. Stufe* (*first-order predicate logic*).

⇝ Nächste Vorlesung!



Zusammenfassung

- Rationale Agenten benötigen *Wissen* über ihre Welt, um rationale Entscheidungen zu treffen.
- Dieses Wissen wird in einer *deklarativen* (Wissensrepräsentations-) Sprache dargestellt und in einer *Wissensbasis* gespeichert.
- Wir benutzen dafür (zunächst) die *Aussagenlogik*.
- Aussagenlogische Formeln können *allgemeingültig, erfüllbar* oder *unerfüllbar* sein.
- Wichtig ist der Begriff der *logischen Folgerbarkeit*.
- Inhaltliches Schlussfolgern kann durch einen *Kalkül* mechanisiert werden $\rightarrow$ Resolution.
- Aussagenlogik wird selbst für kleine Weltausschnitte sehr schnell unhandlich.

Anhang: (Grund-) Resolutions-Theorem (1)

(Grund-)Resolutions-Theorem: Δ ist inkonsistent gdw. $\Delta \vdash \square$.

Beweisidee:

- $\Delta \vdash \square$ impliziert Δ *ist inkonsistent*: wegen Korrektheit der Grundresolvente!
- Δ *ist inkonsistent* impliziert $\Delta \vdash \square$:
  - Sei $V = v_1, \ldots, v_m$ die (Atom-)Menge aller Variablen der Klauselnmenge Δ.
  - Ein vollständiger semantischer Baum B für Δ ist ein Binärbaum der Tiefe m, dessen Kanten der Ebene k vom Vorgängerknoten ausgehend mit den Literalen v_k bzw. $\neg v_k$ markiert sind.
  - → Ein Pfad von der Wurzel bis zur Ebene k entspricht somit einer i.A. partiellen Interpretation $\mathcal{I}$ der Variablen $v_1, \ldots, v_k$ mit $v_i = T$, wenn v_i im Pfad bzw. $v_i = F$, wenn $\neg v_i$ im Pfad.
  - Die Knoten von B sind unmarkiert oder stehen für Klauseln aus Δ .
  - $\mathcal{I}(n)$ bezeichne die Interpretation, die dem Pfad von der Wurzel bis zum Knoten n entspricht.
  - Ein Knoten n heißt *Fehlerknoten*, wenn n der erste Knoten im Pfad ist, so dass $\mathcal{I}(n)$ eine Klausel C aus falsifiziert. n ist dann mit C markiert.

Anhang: (Grund-) Resolutions-Theorem (2)

- Ein semantischer Baum heißt *abgeschlossen*, wenn jeder Pfad mit einem Fehlerknoten endet.
- Ein Knoten n heißt *Inferenzknoten*, wenn *beide* Nachfolgeknoten Fehlerknoten sind.
- Beh.: Δ ist inkonsistent gdw. zu B ein abgeschlossener Teilbaum B' existiert.
- Bew. durch Induktion über Knotenzahl z des semantischen Baums B:
  - Induktionsanfang: $z = 1$: B hat nur Wurzelknoten, also $\square \in \Delta$.
  - Induktionsschritt: $z > 1$:
    - Es existiert mind. ein Inferenzknoten, sonst hätte jeder Knoten mind. einen Nichtfehlerknoten als Nachfolger und damit gäbe es mind. eine erfüllende Interpretation I von Δ
    - Bilde Resolvente aus den durch die beiden Fehlerknoten falsifizierten Clausen.
    - Entferne Nachfolger des Inferenzknotens aus B und mache Inferenzknoten zu neuem Fehlerknoten, da dieser nun die neue Resolvente falsifiziert.
    - Führe diesen Prozess weiter bis zur Reduktion von B zum Wurzelknoten!

Anhang: Beispiel für semantischen Baum

Δ umfasse 4 Klauseln:

C1:	+P	
C2:	+Q	+R
C3:	-P	-Q
C4:	-P	-R