

AI

Kunstmatige Intelligentie (AI)

Hoofdstukken 7/8 van Russell/Norvig = [RN]
Logisch(e) redenerende agenten

voorjaar 2011 — College 9, 5 april 2011

www.liacs.nl/home/kosters/AI/

Eerst bekijken we eenvoudige propositie-logica voor agenten ([RN], Hoofdstuk 7, p. 234–252 en 265–267):

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

dit voor een eenvoudige “Wumpus-wereld”.

Daarna gaan we over op eerste orde (predicaten-)logica ([RN], Hoofdstuk 8, p. 285–306):

$$\forall s \text{ Breezy}(s) \Leftrightarrow \exists r \text{ Adjacent}(r, s) \wedge \text{Pit}(r)$$

Een **Knowledge base** (*KB*) is een verzameling zinnen (“sentences”) in een formele taal. Een *KB* is domein-specifiek. Een **Inference engine** gebruikt logisch goed-gefundeerde algoritmen om met behulp van de *KB* vragen te beantwoorden.

In de **declaratieve** aanpak vertel (“Tell”) je zinnen aan de *KB*, en vraag (“Ask”) je queries. Bij de **procedurele** aanpak vertaal je gewenst gedrag rechtstreeks in programma-code.

Je kunt op verschillende nivo's naar “knowledge based agenten” kijken:

knowledge (kennis) nivo de Niels Bohrweg verbindt Wassenarseweg en Einsteinweg

logisch nivo link (Niels Bohrweg, Wass'weg, Einsteinweg)

implementatie nivo ergens een 1 in een file

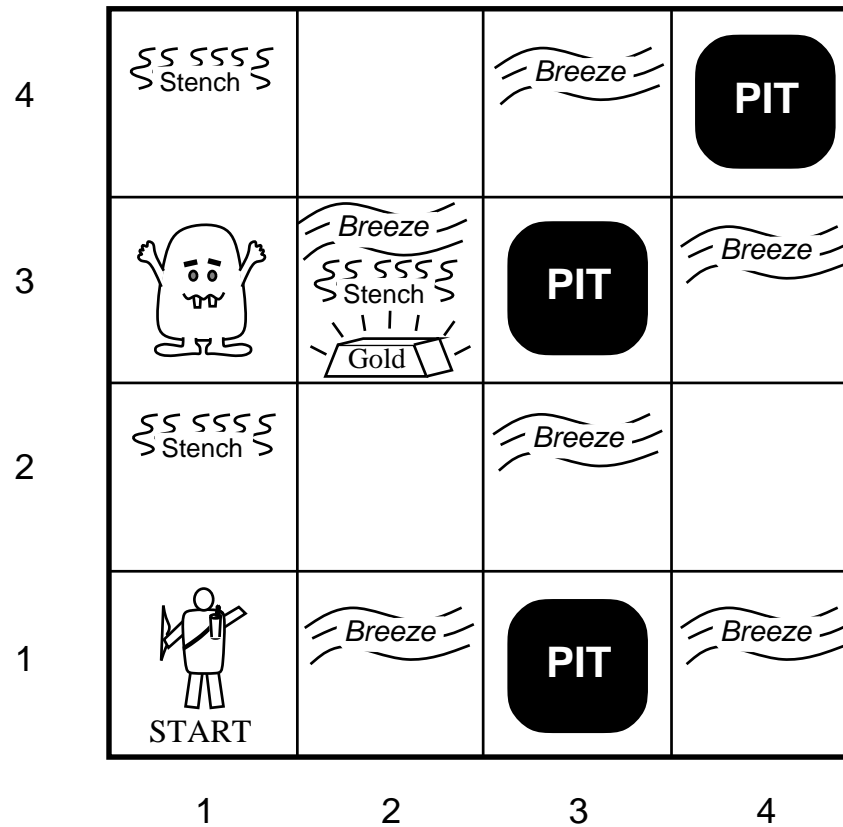
Een eenvoudige kennis-gebaseerde agent zet als volgt een *percept* (waarneming) om in een *actie*, met Knowledge base *KB* en tijd *t* (initieel 0):

```
function KBagent(percept)
  Tell(KB, MakePerceptSentence(percept, t))
  actie ← Ask(KB, MakeActionQuery(t))
  Tell(KB, MakeActionSentence(actie, t))
  t ← t + 1
  return actie
```

De *KB* kan aan het begin **achtergrondkennis** bevatten.

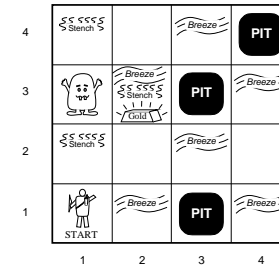
AI—Logische agenten **Wumpus-wereld** — introductie

Als eenvoudig voorbeeld bekijken we de **Wumpus wereld**:



De PEAS beschrijving van de Wumpus-wereld:

Performance maat: goud +1000, dood −1000,
−1 per stap, −10 voor gebruik pijl



Environment: 4×4 , start in (1, 1), random goud, naast (horizontaal, verticaal) Wumpus stinkt het (“stench”), naast put waait het, in het goudvakje glimt het, je kunt de Wumpus doden als je hem “ziet”, slechts één pijl, je kunt goud oppakken in het goudvakje en loslaten waar je wilt, . . .

Actuatoren: TurnLeft, TurnRight, Forward, Grab, Release, Climb (uit de wereld), Shoot

Sensoren: Stench/Smell, Breeze, Glitter, Bump (tegen een muur) en Scream (dode Wumpus) — vijf stuks

De Wumpus-wereld is:

- niet observeerbaar (alleen lokaal),
- deterministisch (uitkomsten liggen vast),
- niet episodisch (sequentieel op het nivo van de acties),
- statisch (Wumpus, goud en putten staan stil),
- discreet en
- single-agent (de Wumpus hoort bij de omgeving).

In het begin, na percept $[-, -, -, -, -]$ ($- = None$), en als je naar $(2, 1)$ bent gegaan, met percept $[-, Breeze, -, -, -]$:

OK			
A OK	OK		

OK	P?		
V OK	A B OK	P?	

A = Agent; OK = veilige plek; V = bezocht; P = put;
B = Breeze; S = Stench; **W** = Wumpus; G = Glitter

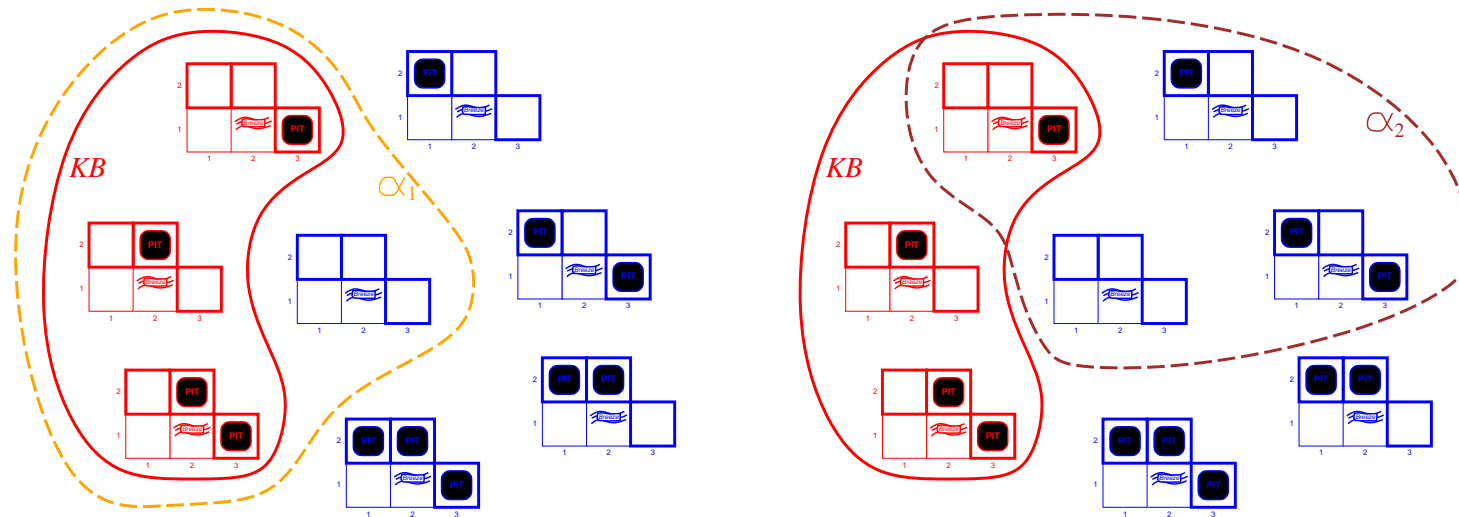
Twee stappen later, met percept $[Stench, -, -, -, -]$, in (1, 2) (via (1, 1)), en als je in stap vijf via (2, 2) naar (2, 3) bent gegaan, met percept $[Stench, Breeze, Glitter, -, -]$:

W			
S A OK	OK		
V OK	B V OK	P	

	P?		
W	A G S B	P?	
S V OK	V OK		
V OK	B V OK	P	

A = Agent; OK = veilige plek; V = bezocht; P = put;
B = Breeze; S = Stench; W = Wumpus; G = Glitter

In de tweede situatie hebben we eerder niets in (1, 1) waargenomen en nu Breeze in (2, 1). We gaan alle “**modellen checken**”, voor wat betreft naburige putten.



Er geldt dat $KB \models \alpha_1$, met $\alpha_1 = \text{“Er is geen put in (1, 2)”}$, maar $KB \not\models \alpha_2$, met $\alpha_2 = \text{“Er is geen put in (2, 2)”}$.

Een eenvoudige logica is de **propositie-logica**.

Syntax:

Sentence \rightarrow AtomicSentence | ComplexSentence

AtomicSentence \rightarrow **True** | **False** | P | Q | R | ...

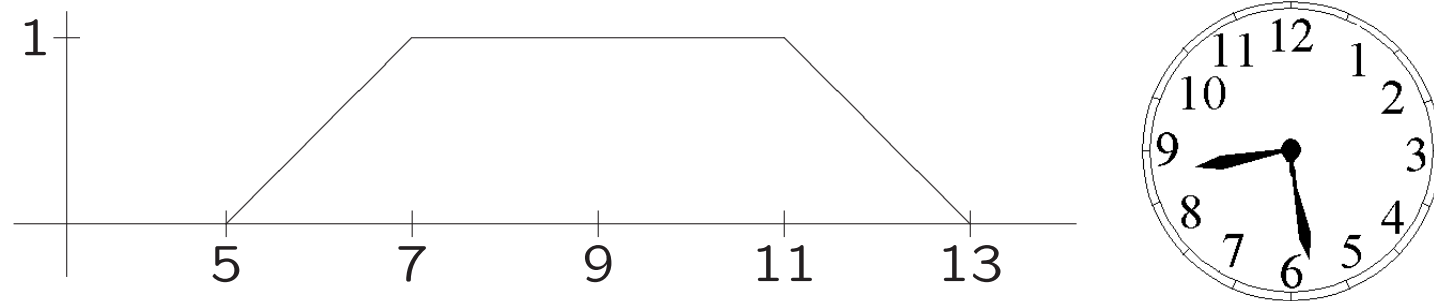
ComplexSentence \rightarrow (Sentence) |

Sentence Connective Sentence | \neg Sentence

Connective \rightarrow \wedge | \vee | \Rightarrow | \Leftrightarrow

Semantiek: de bekende waarheidstafels voor $\neg P$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$ en $P \Leftrightarrow Q$.

Ook mogelijk is **fuzzy logic**, waarbij in plaats van **False/True** de variabelen reële waarden tussen 0 en 1 krijgen:



Hierboven staat het predicaat *ochtend* met bijvoorbeeld:

$$\text{ochtend}(6) = 0.5 \text{ en } \text{ochtend}(8) = 1.0.$$

(En verder heb je ook nog **temporele logica**.)

Voor de eenvoudige propositie-logica gelden de “gewone” inferentieregels, zoals:

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta} \quad (\text{modus ponens}),$$

$$\frac{\alpha \vee \beta, \neg \beta}{\alpha} \quad (\text{unit resolutie}),$$

AND-eliminatie, AND-introductie, OR-introductie, dubbele ontkenning, resolutie, . . .

De stappen uit de inferentie vormen een bewijs. Als je ware (\models , “**entailment**”, “volgen uit”) beweringen afleidt (\vdash), is de inferentie “sound”; als je alle ware beweringen kunt afleiden is de inferentie “compleet”. Zie college Logica.

Voor de Wumpus-wereld introduceren we een grote hoeveelheid logische variabelen, zoals $S_{1,2}$: er is Stench in (1, 2); en $\neg W_{1,2}$ betekent: er staat geen Wumpus op (1, 2).

En regels als:

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

$$R_2 = \quad S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$$

enzovoorts.

Na de Wumpus of Gold gevonden te hebben moet je deze kennis overigens ook nog in acties vertalen.

Hoe vind je de Wumpus?

1. Modus ponens op $\neg S_{1,1}$ en $R_1 = \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$ levert $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. AND-eliminatie: $\neg W_{1,1} \neg W_{1,2} \neg W_{2,1}$
3. Modus ponens op $\neg S_{2,1}$ en $\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$ en daarna AND-eliminatie geeft: $(\neg W_{1,1})$
 $(\neg W_{2,1}) \neg W_{2,2} \neg W_{3,1}$
4. Modus ponens op $S_{1,2}$ en $R_2 = S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$ geeft: $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
5. Unit resolutie met $\alpha = W_{1,3} \vee W_{1,2} \vee W_{2,2}$ en $\beta = W_{1,1}$ levert: $W_{1,3} \vee W_{1,2} \vee W_{2,2}$
6. Unit resolutie met $\alpha = W_{1,3} \vee W_{1,2}$ en $\beta = W_{2,2}$ levert: $W_{1,3} \vee W_{1,2}$
7. Unit resolutie met $\alpha = W_{1,3}$ en $\beta = W_{1,2}$ geeft: $W_{1,3}$

Er zijn allerlei regels, zoals

$$W_{1,1} \vee W_{2,1} \vee \dots \vee W_{4,4}$$

(er is minstens één Wumpus), en $\neg W_{2,3} \vee \neg W_{3,4}$ enzovoorts om maximaal één Wumpus af te dwingen. En een Wumpus stinkt, dus voor alle $1 \leq x, y \leq 4$ hebben we

$$S_{x,y} \Leftrightarrow W_{x,y} \vee W_{x,y-1} \vee W_{x,y+1} \vee W_{x+1,y} \vee W_{x-1,y}$$

(16 regels; langs de randen iets anders) — en ook zoiets voor het verband tussen Breeze en putten.

Nu moet je ook nog bijhouden waar je bent:

$$L_{1,1}^0 \wedge \text{FacingEast}^0 \wedge \text{Forward}^0 \Rightarrow L_{2,1}^1 \wedge \neg L_{1,1}^1$$

(met tijd-superscript; een “effect-axioma”, zie later).

Er zijn dus allerlei problemen met deze eenvoudige predi-
caten-logica:

- zo krijg je veel te veel proposities (je mag geen varia-
belen als index gebruiken) en
- zijn er problemen met de tijd, die je eigenlijk ook als
index wilt gebruiken ($L_{1,1}^0$: de agent is op tijdstip 0 op
positie (1, 1)).

Dus gaan we over naar eerste orde (predicaten-)logica, met
objecten en relaties.

Een krachtiger logica is de **eerste orde logica**, met syntax:

Sentence \rightarrow AtomicSentence | \neg Sentence

| Sentence Connective Sentence | (Sentence)

| Quantifier Variable, . . . Sentence

AtomicSentence \rightarrow Predicate(Term, . . .) | Term = Term

Term \rightarrow Function(Term, . . .) | Constant | Variable

Quantifier \rightarrow \forall | \exists

Connective \rightarrow \wedge | \vee | \Rightarrow | \Leftrightarrow

Constant \rightarrow A | X_1 | John | . . .

Variable \rightarrow a | x | s | . . .

Predicate \rightarrow Before | HasColor | . . .

Function \rightarrow Mother | LeftLegOf | . . .

Enkele voorbeeldzinnen in eerste orde logica:

$$\forall x \text{ Cat}(x) \Rightarrow \text{Mammal}(x)$$

$$\forall x [\text{Cat}(x) \vee (\exists x \text{ Brother}(\text{Richard}, x))]$$

In deze laatste kun je de binnenste x beter door een y vervangen.

Een term zonder variabelen heet wel een **grond-term**.

In **Prolog** worden de constanten juist met kleine letters geschreven, en de variabelen met hoofdletters: `cat(felix)`.
(grond-term) en `cat(X) :- mammal(X),furry(X)`.

Er zijn ook **hogere orde logica**'s, waarin je bijvoorbeeld over functies kunt quantificeren/redeneren:

$$\forall f, g (f = g) \Leftrightarrow (\forall x f(x) = g(x))$$

(Overigens is “=” een predicaat Is.)

De speciale quantifier “**∃!**”: $\exists!x \text{ King}(x)$ is equivalent met $\exists x \text{ King}(x) \wedge \forall y (\text{King}(y) \Rightarrow x = y)$

En dan heb je nog de **λ-calculus**.

Een λ-expressie als $\lambda x, y x^2 - y^2$ werkt als volgt:

$$(\lambda x, y x^2 - y^2)(25, 24) = 25^2 - 24^2 = 49$$

Doorgaans gebruiken we eerste orde logica in een geschikt beperkt **domein**, bijvoorbeeld *familie*. Redelijke basis-predicaten zijn in dat geval Female, Spouse en Parent.

Je krijgt dan regels als:

$$\forall m, c \text{ Mother}(m, c) \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c)$$

$$\forall w, h \text{ Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w)$$

$$\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x)$$

$$\forall p, c \text{ Parent}(p, c) \Leftrightarrow \text{Child}(c, p)$$

$$\forall g, c \text{ Grandparent}(g, c) \Leftrightarrow \exists p (\text{Parent}(g, p) \wedge \text{Parent}(p, c))$$

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow x \neq y \wedge \exists p (\text{Parent}(p, x) \wedge \text{Parent}(p, y))$$

Dit zijn **axioma**'s; er zijn ook **stellingen**, zoals

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x)$$

We kunnen nu beter dan met de eenvoudige propositie-logica onze Wumpus-wereld beschrijven.

Een “percept-rijtje” op tijdstip 5 ziet er uit als:

$Percept([Stench, Breeze, Glitter, None, None], 5)$.

We “Ask”-en een query als $\exists a BestAction(a, 5)$, en verwachten een lijstje als $\{a/Grab\}$ met “bindingen”.

Er zijn eenvoudige “perceptie-regels”, zoals:

$\forall t, s, m, g, c Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$

En “reflex-gedrag” komt voort uit:

$\forall t Glitter(t) \Rightarrow BestAction(Grab, t)$

Hoe representeer je de omgeving?

Naast (tot en met hier toe) **synchrone** regels zijn er overigens ook **diachrone** (redeneren met tijd erbij; zie later).

Je krijgt regels als:

$$\begin{aligned} \forall x, y, a, b \text{ } Adjacent((x, y), (a, b)) &\Leftrightarrow \\ (a, b) \in \{(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)\} & \\ \forall s, t \text{ } At(Agent, s, t) \wedge Breeze(t) &\Rightarrow Breezy(s) \end{aligned}$$

met coördinaten x, y, a, b , vakje s en tijd t ; een vakje is een coördinatenpaar. *Breezy* hangt niet van t (tijd) af.

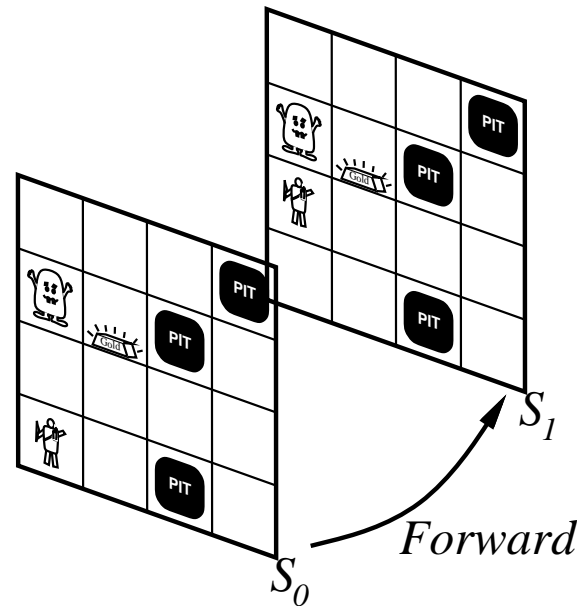
Diagnostische regels leiden van waargenomen effecten naar verborgen oorzaken (“koorts \Rightarrow ziek”), **causale regels** net andersom (“ziek \Rightarrow koorts”):

$$\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Adjacent}(r, s) \wedge \text{Pit}(r) \quad (\text{diagnostisch})$$

$$\forall r \text{ Pit}(r) \Rightarrow [\forall s \text{ Adjacent}(r, s) \Rightarrow \text{Breezy}(s)] \quad (\text{causaal})$$

Systemen die redeneren met causale regels heten wel **model-based reasoning** systemen.

Situaties zijn: de beginsituatie, en dat wat je krijgt als je een actie op een situatie loslaat. De benadering die probeert veranderingen bij te houden heet **situation calculus**.



Bepaalde “eigenschappen” zijn niet afhankelijk van de tijd, en zijn **eeuwig** of **atemporeel**: $Wall((0, 1))$.

Effect-axioma's beschrijven veranderingen ten gevolge van acties (met situaties s):

$$\forall s \text{ AtGold}(s) \Rightarrow \text{Holding}(\text{Gold}, \text{Result}(\text{Grab}, s))$$

Frame-axioma's beschrijven wat onveranderd blijft tijdens acties (met voorwerpen x en acties a):

$$\forall a, x, s \text{ Holding}(x, s) \wedge a \neq \text{Release} \Rightarrow \text{Holding}(x, \text{Result}(a, s))$$

En samen in **successor-state axioma's**:

$$\forall a, s \text{ Holding}(\text{Gold}, \text{Result}(a, s)) \Leftrightarrow ((a = \text{Grab} \wedge \text{AtGold}(s)) \vee (\text{Holding}(\text{Gold}, s) \wedge a \neq \text{Release}))$$

Dit lost het zogeheten **representational frame problem** op (te veel dingen waar een actie niets mee doet).

Het is nog weer een verhaal apart hoe je **plannen** maakt.

En hoe je het **inferential frame problem** aanpakt.

En hoe je **voorkeuren** verwoordt.

En dan zijn er nog **possibility axioma's**.

Enzovoorts.

Het huiswerk voor de volgende keer (dinsdag 12 april 2011): lees **Hoofdstuk 4.1.4**, p. 126–129 over Genetische algoritmen, eens door.

Denk na over de derde opgave: **Neurale Netwerken** — het is dan namelijk de deadline!

Werkcollege over sommen: maandag 11 april 2011, 13.45–15.30 uur, zaal 409, zie

<http://www.liacs.nl/home/kosters/AI/opgaven3.pdf>