Kunstmatige Intelligentie (AI)

Hoofdstuk 2 van Russell/Norvig = [RN]

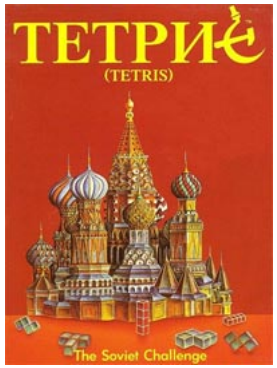
Intelligente agenten

voorjaar 2021

College 2, 8 februari 2021

www.liacs.leidenuniv.nl/~kosterswa/AI/agenten.pdf

Een paar opmerkingen over Tetris:



- Video's; C++ en voorbeeldcode
- En de verschillende strategieën? Monte Carlo en "ook slim"?
- Experimenten; grafieken, statistieken.
- Het verslag (in LaTeX).

www.liacs.leidenuniv.nl/~kosterswa/AI/teet2021.html

Een agent is iets wat zijn/haar omgeving (environment) waarneemt met behulp van sensoren en op deze omgeving werkt met actuatoren.
Een mens in de fysieke wereld heeft ogen en handen.

De waarnemingen die een agent gedaan heeft vormen zijn of haar percept sequence (rij met waarnemingen). De door de agent te kiezen actie kan in principe van die hele rij afhangen, maar niet op iets wat niet is waargenomen.
De agent functie beeldt rijtjes af op acties.

Wat rationeel is hangt af van:

- de performance maat (measure) die de mate van succes definieert;
- wat de agent weet van zijn/haar omgeving (voorkennis = "prior knowledge");
- de mogelijke acties;
- de percept sequence op dat moment.

Een ideale rationele agent moet voor iedere mogelijke percept sequence die actie uitvoeren waarvan verwacht wordt dat deze de performance maat maximaliseert, op basis van dat rijtje en alle ingebouwde kennis die de agent bezit.
NB Dit is iets anders dan alwetend zijn.

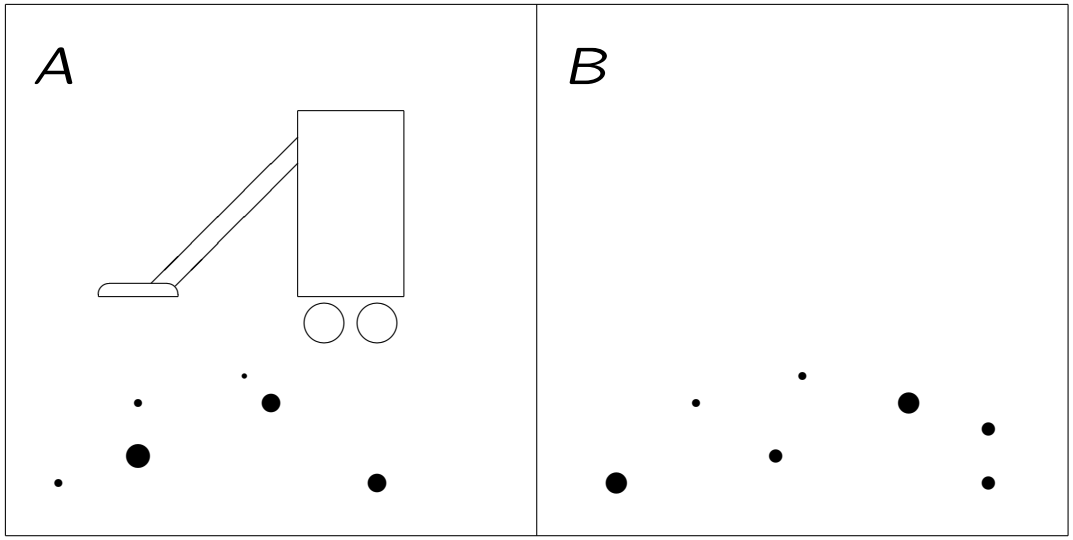
De ideale afbeelding van percept sequences naar acties specificeert welke actie de agent zou moeten nemen, gegeven een percept sequence. Dit levert een ideale agent.

Een agent is autonoom als zijn/haar gedrag wordt bepaald door zijn/haar *eigen* ervaring.

Als er geen aandacht is voor percepts is er geen sprake van autonomie.
Als een agent autonoom is, vertoont hij/zij doorgaans *lerend* gedrag — om te compenseren voor gedeeltelijke of foutieve voorkennis.

Voorbeeld: een altijd (zomer- en wintertijd) gelijk lopend horloge is niet per se autonoom. En een op afstand bedienbare "robot" al zeker niet.

De Stofzuiger-wereld ziet er als volgt uit:



Mogelijke acties: *Left* (stukje naar links), *Right* (stukje naar rechts), *Suck*, *Nothing*.
Eén waarneming (oftewel percept) is bijvoorbeeld [*A*,*Clean*] of [*B*,*Dirty*].

Een eenvoudige agent voor de Stofzuiger-wereld zou de volgende tabel kunnen benutten:

Percept sequence	Actie
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
...	...
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
...	...

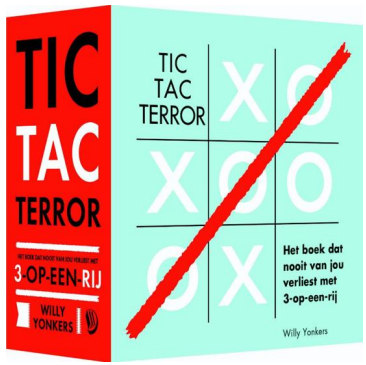
Dit levert een rationele agent op.

NB We nemen aan dat *Left*/*Right* een klein stukje beweegt (als je al helemaal rechts bent, gebeurt er bij *Right* niets).

Opzoektabel

Waarom gebruik je in het algemeen geen opzoektabel, met een complete opsomming van alle mogelijkheden?

- Zelfs voor een "simpele" schaak spelende agent heb je zo'n 35^{100} ($35 \approx$ aantal mogelijke zetten; $100 \approx$ lengte partij) regels nodig. Dus geheugenproblemen.
- Het zou erg lang duren een dergelijke tabel te vullen.
- Er is geen autonomie. Problemen met veranderende omgeving.
- Zelfs met leren erbij zou het oneindig lang (kunnen) duren.
- Boeit niet.



Als we een rationele agent willen ontwerpen, moeten we de task environment (taak-omgeving) specificeren.
Bijvoorbeeld, voor een automatische taxi-chauffeur:

P erformance maat: veiligheid, winst, bestemming, comfort, de wet, . . .

E nvironment (Omgeving): straten, verkeer, voetgangers, weer, . . .

A ctuatoren: stuur, remmen, gaspedaal, scherm, . . .

S ensoren: camera, meters, microfoon, GPS, toetsenbord, . . .

En voor een systeem ten behoeve van medische diagnose:

P erformance maat: gezonde patient, lage kosten, rechtzaken, . . .

E nvironment: patient, ziekenhuis, staf, . . .

A ctuatoren: vragen, onderzoeken, diagnoses, behandelingen, . . .

S ensoren: antwoorden patient, invoeren symptomen op toetsenbord, thermometer, . . .

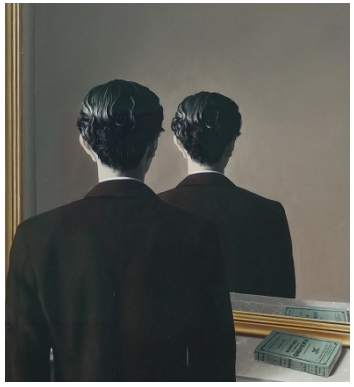
Probeer zelf eens de omgeving voor een agent ten behoeve van internet-winkelen te beschrijven.

Er zijn verschillende dimensies waarlangs je — na veel discussie — omgevingen kunt leggen:

- volledig observeerbaar $\leftrightarrow$ deels observeerbaar
- deterministisch $\leftrightarrow$ niet-deterministisch
  stochastisch $\approx$ niet-deterministisch met kansen
  strategisch: det., afgezien van acties andere agenten
- episodisch $\leftrightarrow$ sequentieel
- statisch $\leftrightarrow$ dynamisch
  semi-dynamisch: alleen score verandert met tijd
- discreet $\leftrightarrow$ continu
- enkele agent $\leftrightarrow$ multi-agent (competitief, cooperatief)

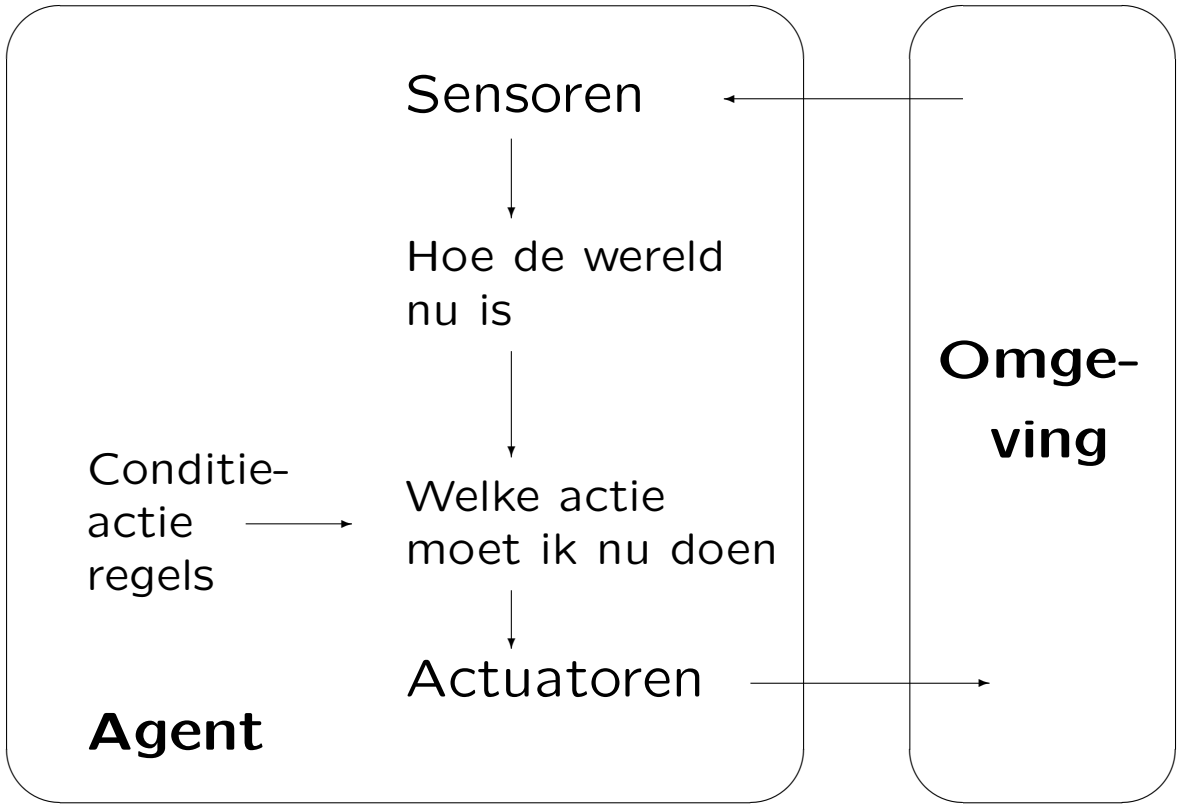
Daarnaast kan er van alles over de omgeving (on)bekend zijn (known $\leftrightarrow$ unknown). Een voorbeeld: ken je de spelregels van poker al, of leer je die tijdens het spelen?

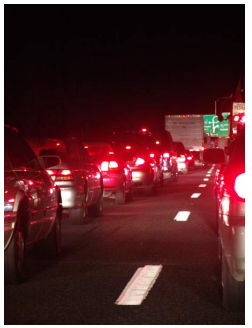
Dit is iets anders dan volledig observeerbaar $\leftrightarrow$ deels observeerbaar! Voorbeeld: bij patience ken je de spelregels, maar weet je niet alle kaarten (deels observeerbaar); bij een nieuw computerspel kun je alles zien, maar weet je soms nog niet hoe de "knoppen" werken (unknown).



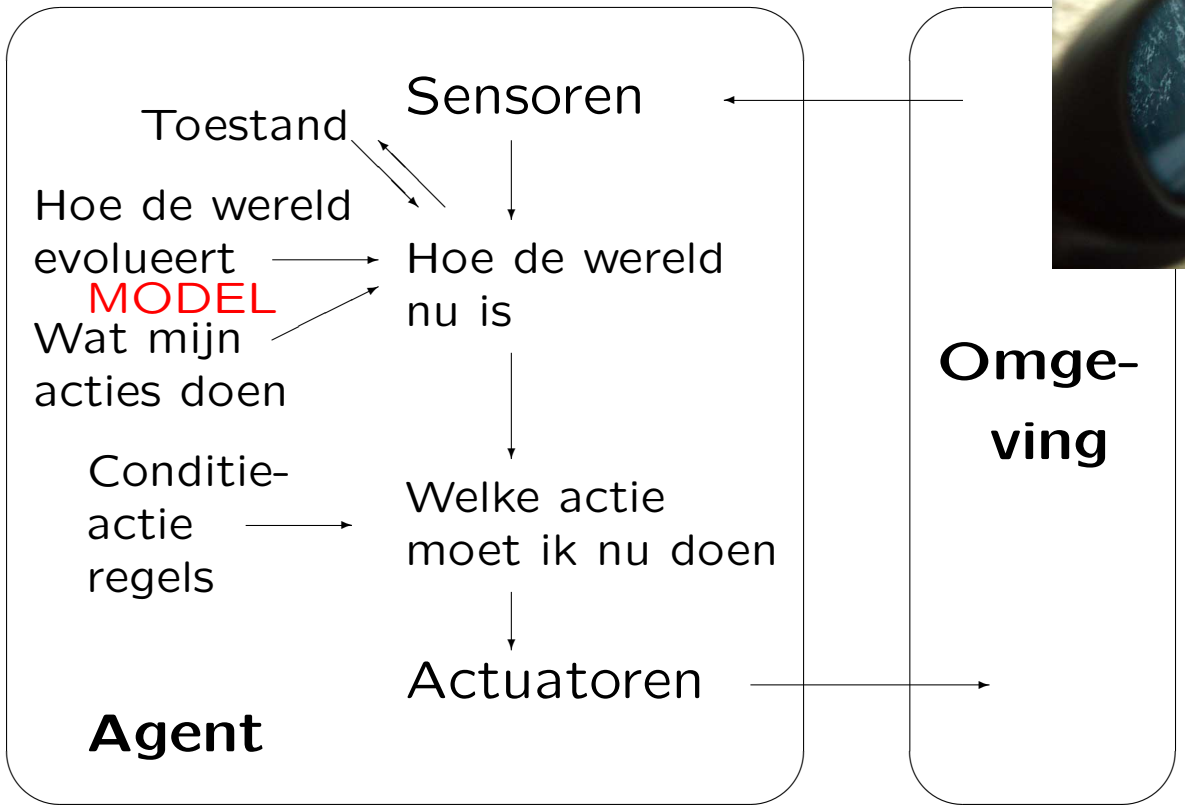
	Obs.	Det.	Epis.	Stat.	Disc.	Agt.
Puzzel	J	J	N	J	J	1
Schaak+klok	J	J	N	semi	J	> 1
Poker	N	J	N	J	J	> 1
Backgammon	J	N	N	J	J	> 1
Taxi rijden	N	N	N	N	N	> 1
Beeldanalyse	J	J	J	semi	N	1
Fabrieks-robot	N	N	J	N	N	1
WWW-winkel	N	$\pm$	N	semi	J	$1/> 1$

Volledig observeerbaar: alle *relevante* zaken gezien.
Deterministisch: volgende toestand wordt bepaald door huidige toestand en actie (gezien vanuit de agent).
Episodisch: onderling onafhankelijke "atomaire" episodes; als voorbeeld: een schaaktoernooi.
Sommige poker-varianten zijn wel stochastisch.



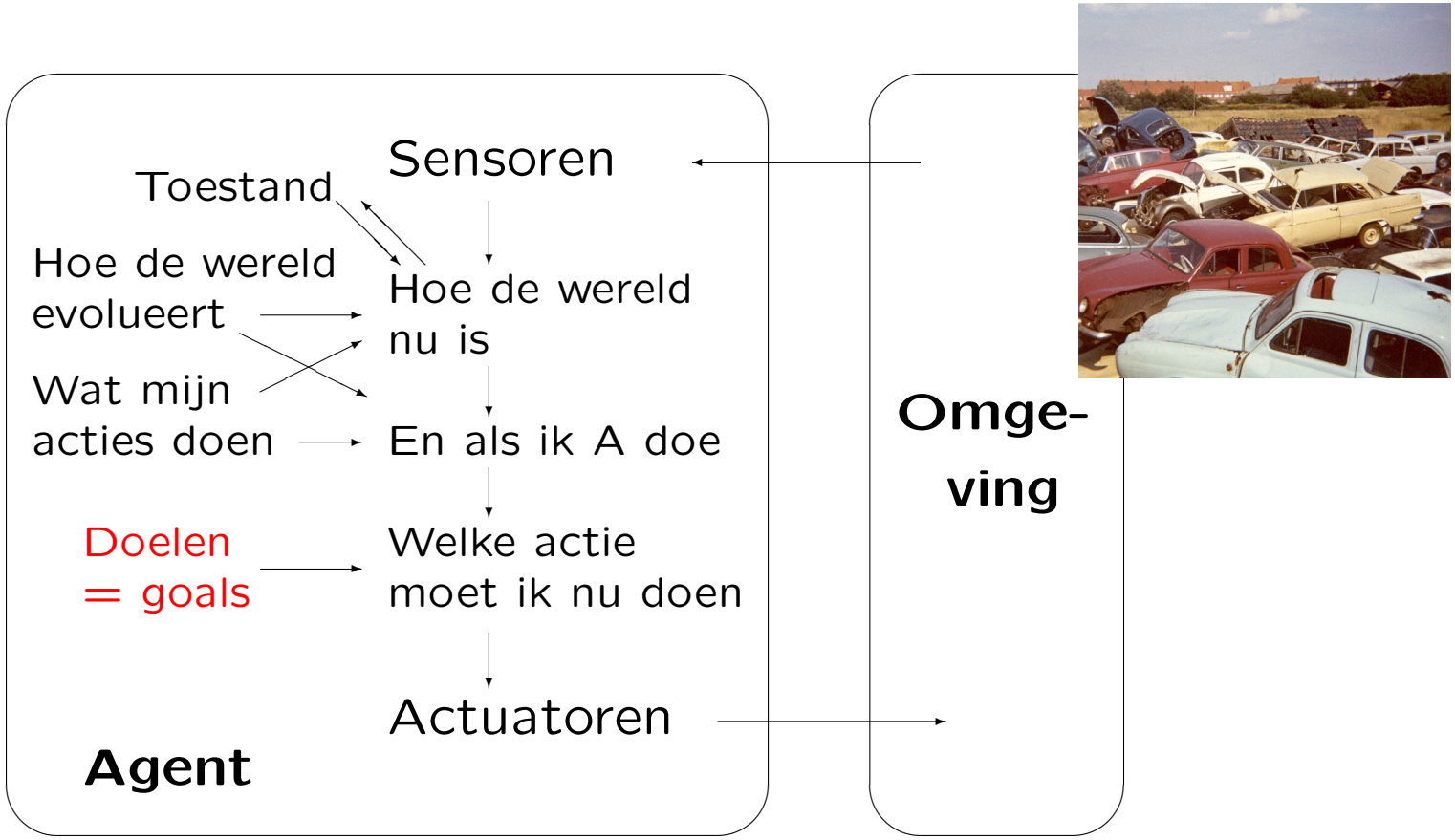


ALS auto_voor_je_remt DAN begin_zelf_te_remmen
Randomiseren helpt je soms uit oneindige loops.

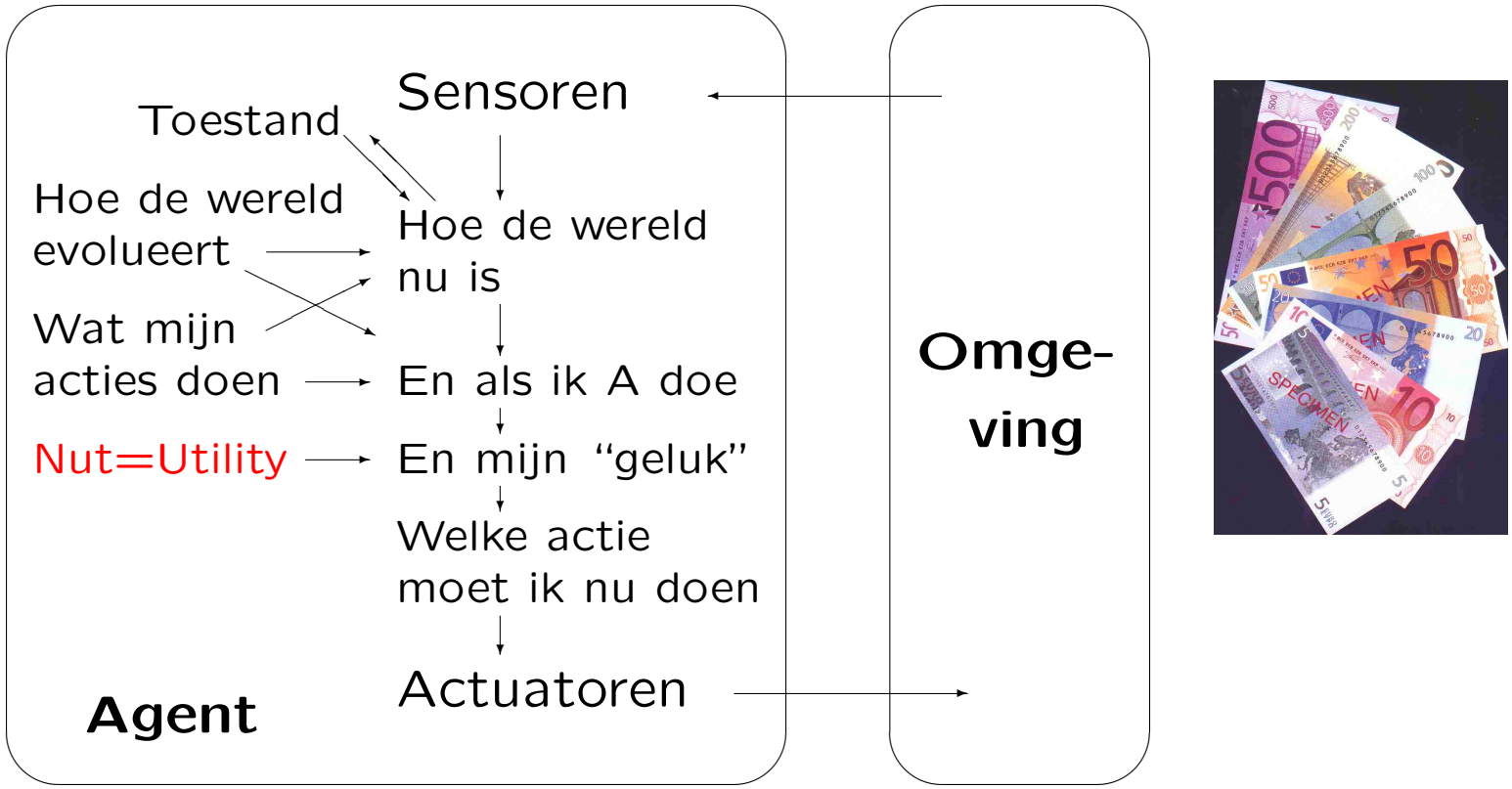


Wat zagen we zo-even in de spiegel?

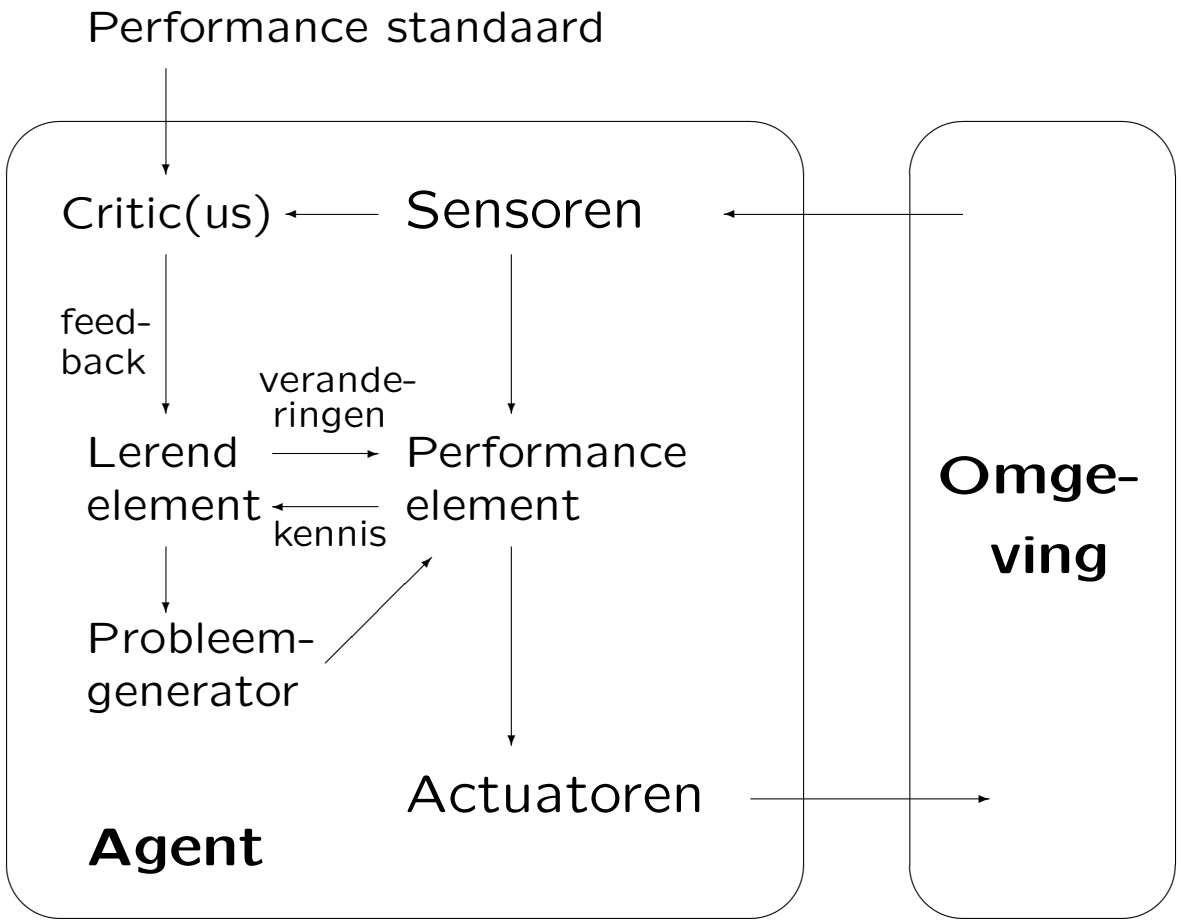
De agent heet wel model-gebaseerd.

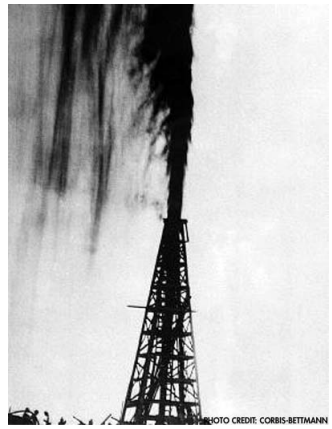


Waar moet de auto naar toe? (ook weer model-gebaseerd)



Hoe snel/veilig/duur/... wordt de bestemming bereikt?
De utility-functie "weegt" doelen en meet kansen.





De probleem-generator geeft exploratie ($\leftrightarrow$ exploitatie).

Het Belief-Desire-Intention model (BDI) is een door de filosoof/psycholoog Michael Bratman in de jaren 80 ontwikkeld schema om praktisch redeneren te kunnen begrijpen en analyseren.

De software-kant van het model bestudeert en programmeert agenten. Het model probeert het selecteren van plannen te scheiden van het uitvoeren daarvan, maar creëert zelf geen plannen.

Een BDI-agent heeft:

beliefs wat gelooft de agent over zichzelf en de wereld?

desires wat wil de agent graag?

een doel (goal) is een verlangen (desire) dat de agent actief najaagt, consistent met andere doelen

intentions wat heeft de agent gekozen?

de agent besluit voor een of meer plannen/acties

Een voorbeeld:

- je gelooft dat op 1 april college AI is
- je wilt graag AI halen, uitslapen, feesten
- je neemt je voor naar ieder college te gaan

Een programma voor een Eenvoudige reflex-agent (al dan niet met een op een model van de wereld gebaseerde toestand) is:

toestand $\leftarrow$ Interpreteer_Input (*percept*)
regel $\leftarrow$ Regel_Match (*toestand*,*regels*)
actie $\leftarrow$ Regel_Actie[*regel*]

En in concreto voor de Stofzuiger:

if *status* = *Dirty* **then return** *Suck*
else if *locatie* = *A* **then return** *Right*
else return *Left*

Je kunt op verschillende nivo's naar componenten van agenten kijken (lees ook Rodney Brooks):

atomair geen interne structuur in de toestanden
zoeken in grafen, spel(l)en, . . .

opgedeeld = **factored** toestanden worden bepaald door variabelen met hun waardes
CSP's, propositie-logica, Bayesiaanse netwerken, . . .

gestructureerd zaken zijn met elkaar gerelateerd
eerste-orde logica, natuurlijke taal, . . .

Het huiswerk voor de volgende keer (15 februari 2021): lees Hoofdstuk 7 en 8, p. 208–224, p. 237–240 en p. 251–271 van [RN] door (in de derde druk p. 234–252, p. 265–267 en p. 285–306), over het onderwerp Logische agenten.

Denk tevens aan de eerste opgave: Monte Carlo & Tetris; deadline: 23 februari 2021.