

AI

Kunstmatige Intelligentie (AI)

Hoofdstuk 2 van Russell/Norvig = [RN]
Intelligente agenten

voorjaar 2012

College 2, 14 februari 2012 (= DH—7 maart 2012)

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

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 specificiert welke actie de agent zou moeten nemen, gegeven een percept sequence. Dit levert een ideale agent.

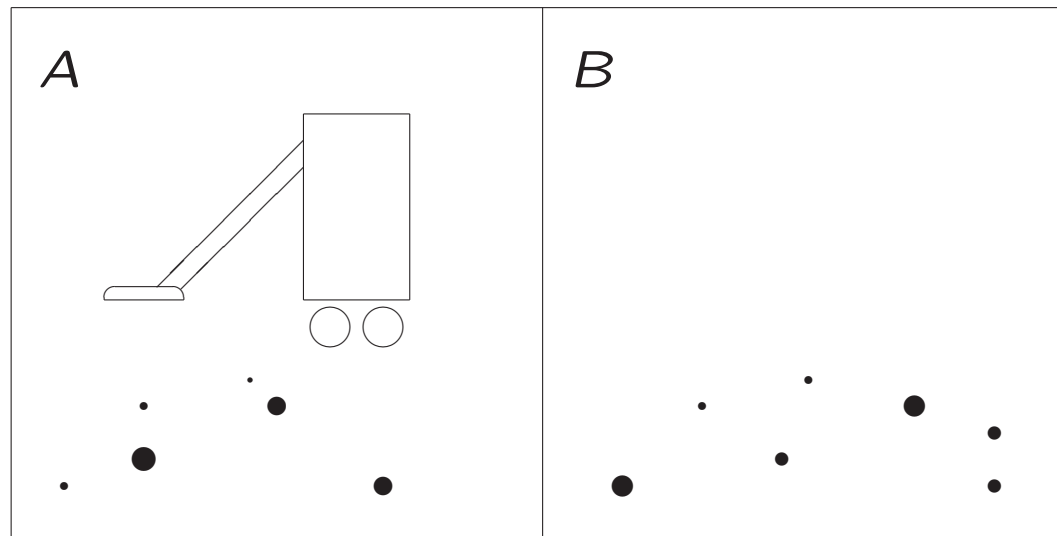
Een agent is **autonoom** als zijn/haar gedrag wordt bepaald door zijn/haar eigen ervaring.

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 overal gelijk lopend horloge is niet per se autonoom. En een op afstand bedienbare “robot” ook 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
$[A, \textit{Clean}]$	<i>Right</i>
$[A, \textit{Dirty}]$	<i>Suck</i>
$[B, \textit{Clean}]$	<i>Left</i>
$[B, \textit{Dirty}]$	<i>Suck</i>
$[A, \textit{Clean}], [A, \textit{Clean}]$	<i>Right</i>
$[A, \textit{Clean}], [A, \textit{Dirty}]$	<i>Suck</i>
...	...
$[A, \textit{Clean}], [A, \textit{Clean}], [A, \textit{Clean}]$	<i>Right</i>
...	...



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 niets).

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, . . .



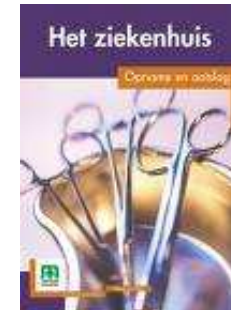
En voor een systeem ten behoeve van medische diagnose:

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

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

A ctuatoren: vragen, onderzoeken, diagnoses, behandel-
lingen, . . .

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



Probeer zelf eens een agent voor internet-winkelen te be-
schrijven.

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

- volledig observeerbaar \leftrightarrow deels observeerbaar
- deterministisch \leftrightarrow stochastisch
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. 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; bij een nieuw computerspel kun je alles zien, maar weet je nog niet hoe de knoppen werken.

NB Bij deterministisch wordt ook bedoeld: afgezien van eventuele acties van andere agenten (*strategisch*).

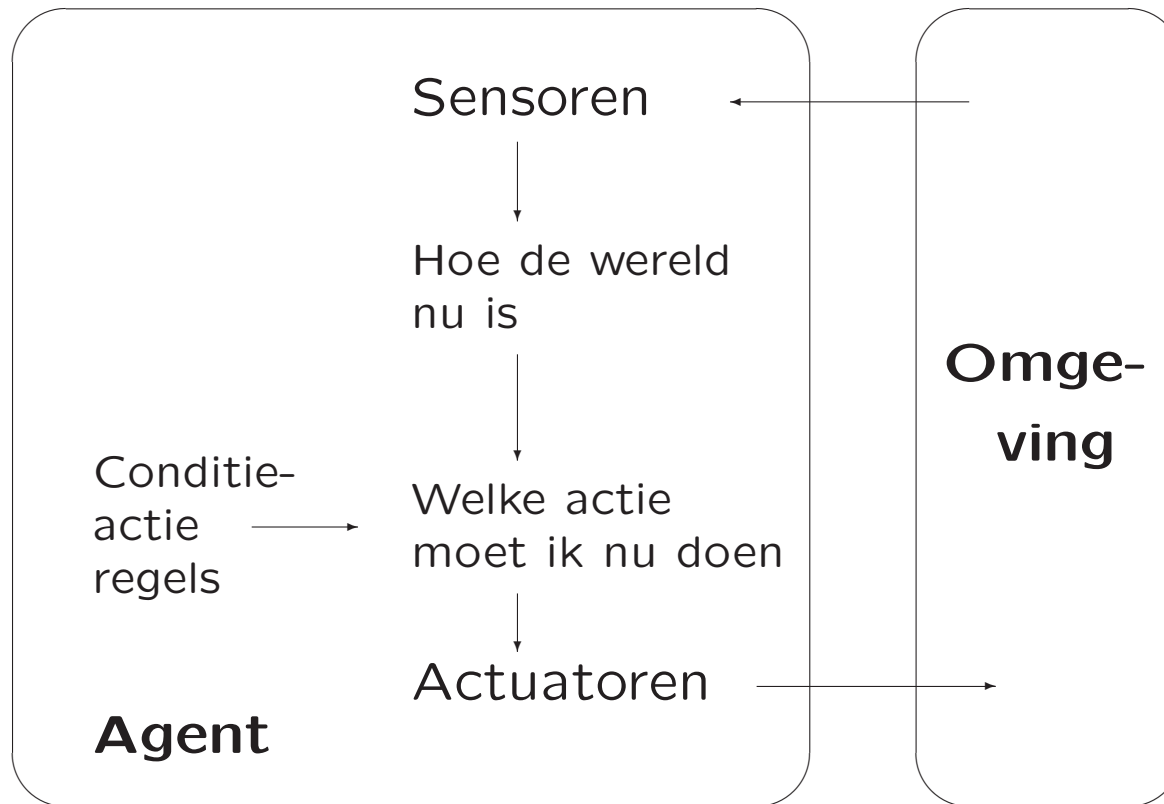
	Obs.	Det.	Epis.	Stat.	Disc.	Agt.
Puzzel	J	J	N	J	J	1
Schaak+klok	J	J	N	semi	J	> 1
Poker	N	J	N	J	J	> 1
Backgammon	J	N	N	J	J	> 1
Taxi rijden	N	N	N	N	N	> 1
Beeldanalyse	J	J	J	semi	N	1
Auto-robot	N	N	J	N	N	1
WWW-winkel	N	±	N	semi	J	1/> 1

Volledig observeerbaar: alle *relevante* zaken gezien.

Deterministisch: volgende toestand bepaald door huidige toestand en actie (gezien vanuit de agent).

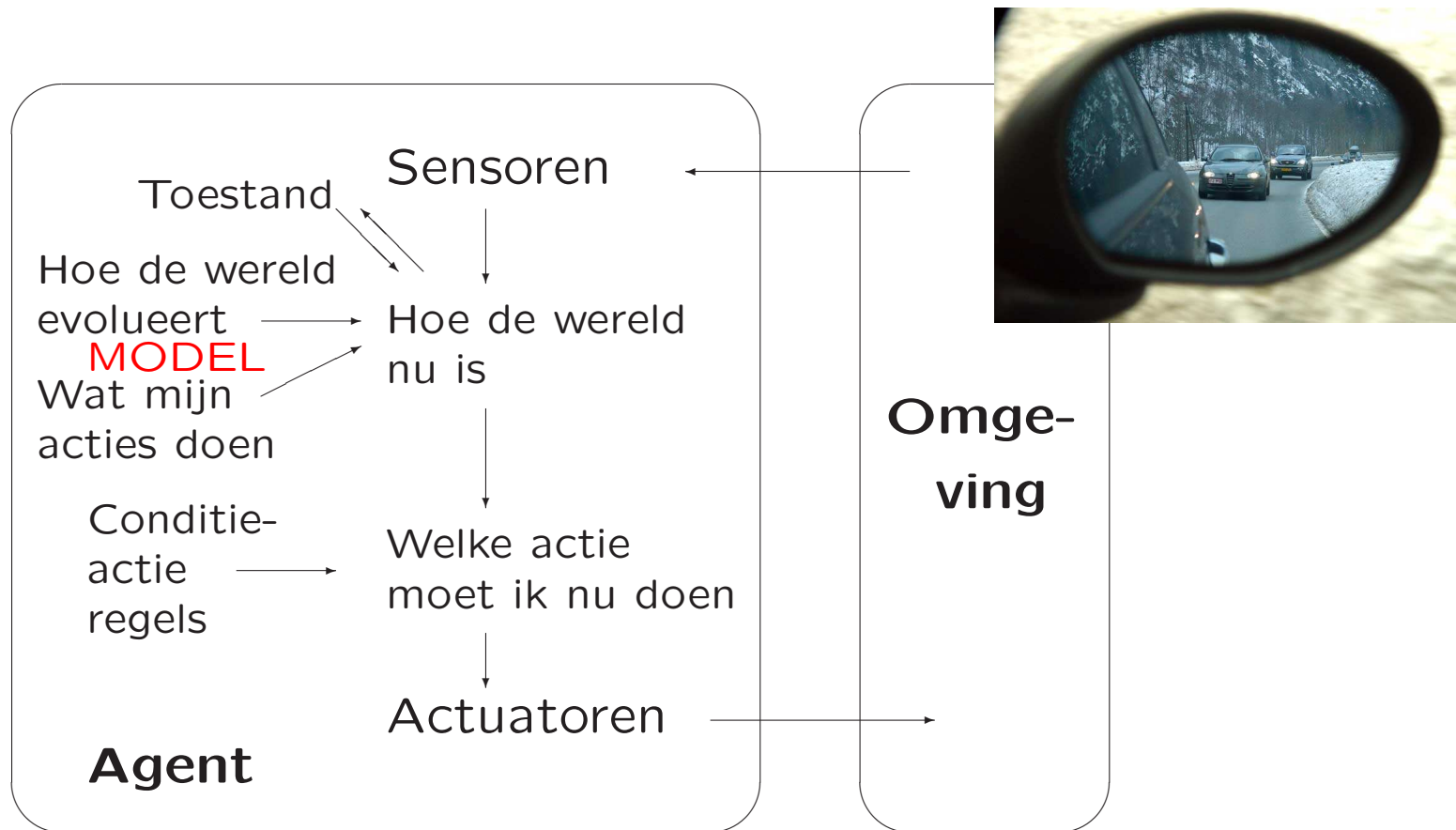
Een schaaktoernooi is episodisch.

Sommige poker-varianten zijn wel stochastisch.

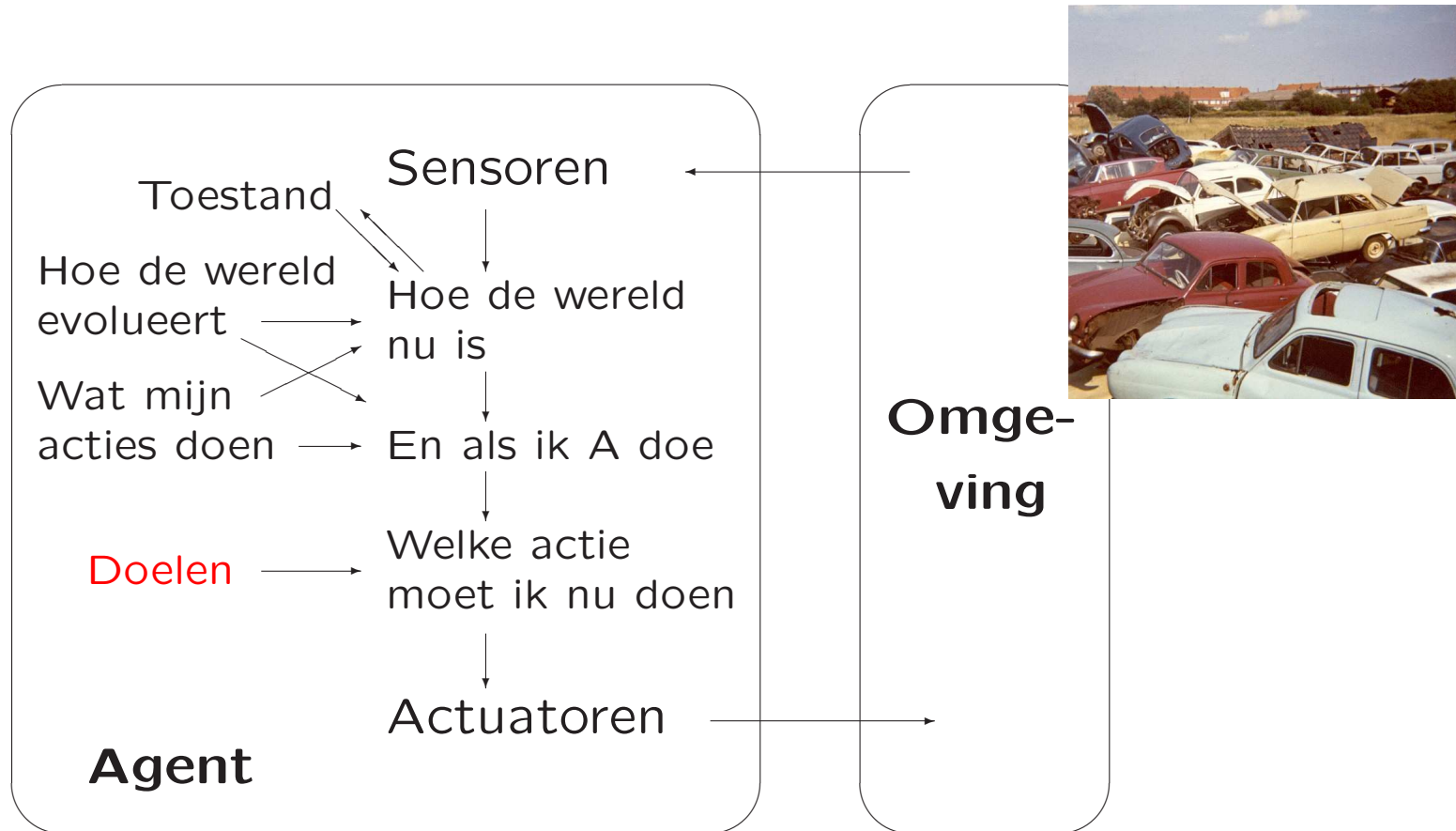


ALS auto_voor_je_remt DAN begin_zelf_te_remmen

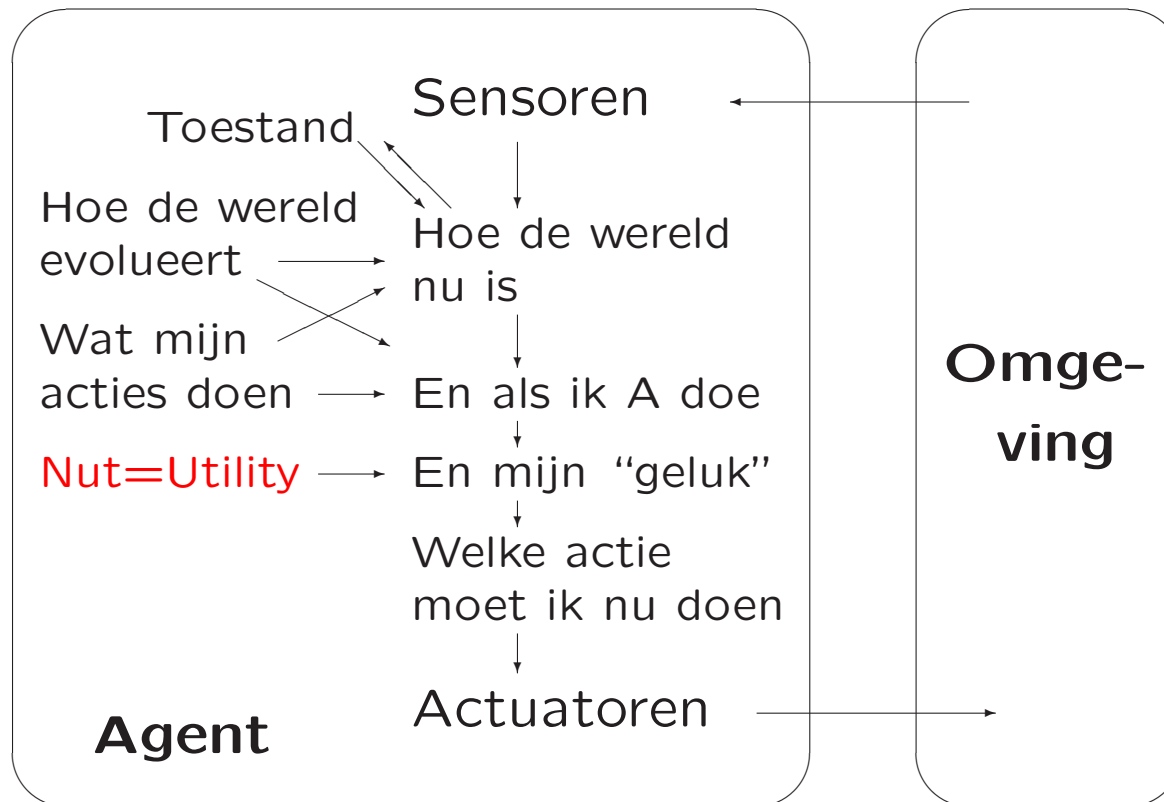
AI—Intelligente agenten **Reflex-agenten met toestand**



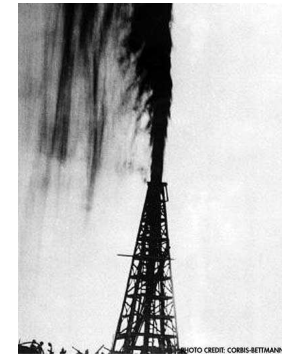
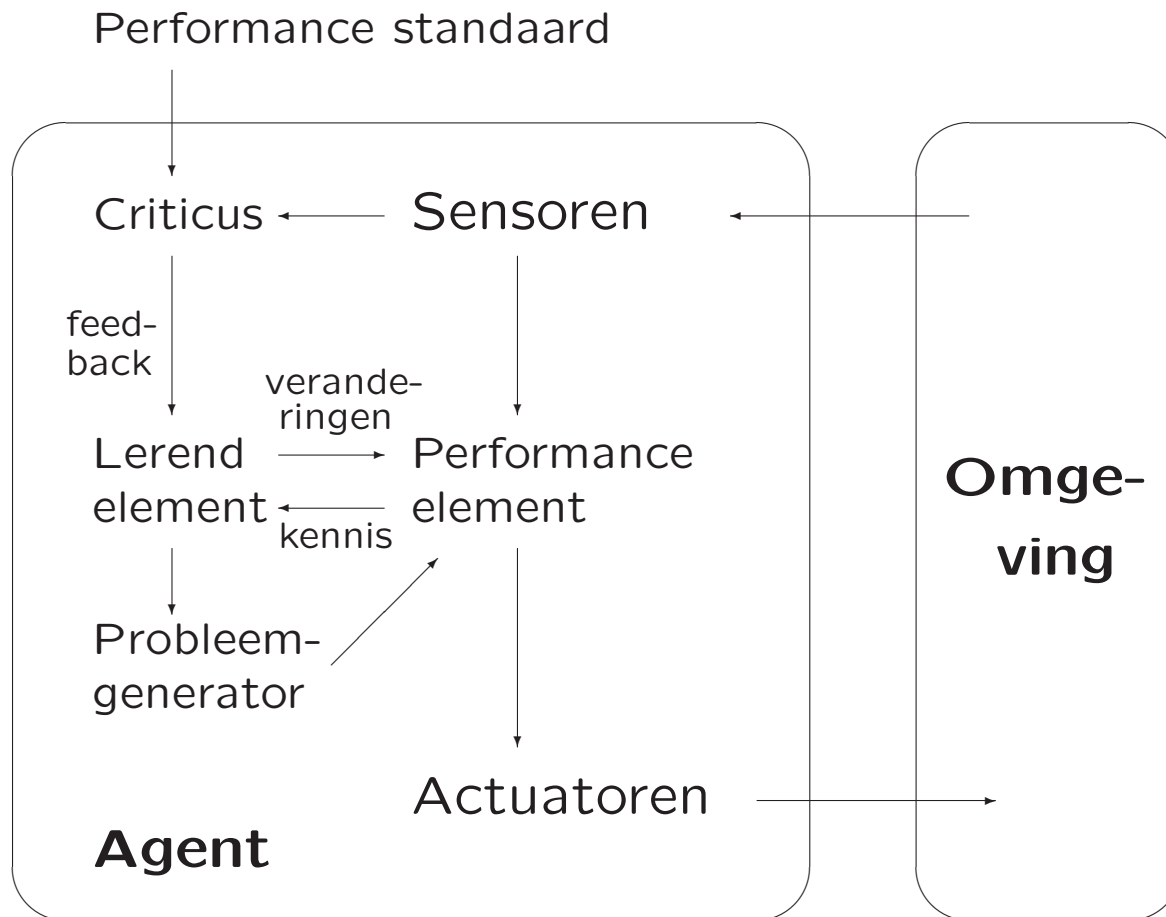
Wat zagen we zo-even in de spiegel?
De agent heet wel **model-gebaseerd**.



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**).

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

```
toestand ← Interpreteer_Input (percept)  
regel ← Regel_Match (toestand, regels)  
actie ← 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:

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

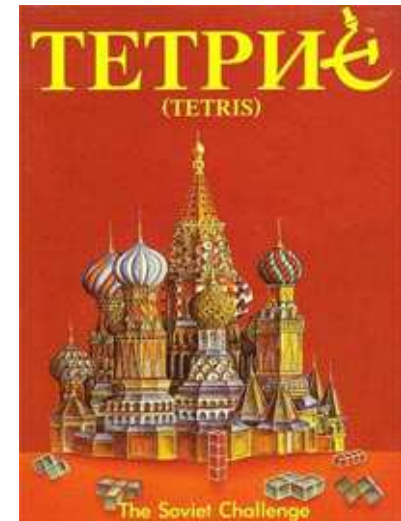
opgedeeld = “**factored**” toestanden worden bepaald door
variabelen met hun waardes
CSP's, propositie-logica, Bayesiaanse netwerken, . . .

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

Het huiswerk voor de volgende keer (dinsdag 21 februari 2012 / DH: woensdag 14 maart 2012): lees **Hoofdstuk 3**, p. 64–119 van [RN] door. Kijk ook eens naar de vragen bij dit hoofdstuk.

Denk tevens aan de eerste opgave: **Tetris**;
deadline: 28 februari / DH: 21 maart 2012.

Denk aan de “vragenuren” (na elk college).



Een paar opmerkingen over Tetris:

- Zijn de spelregels duidelijk?
- En de verschillende strategieën?
- Experimenten.
- Het verslag (in \LaTeX).

