

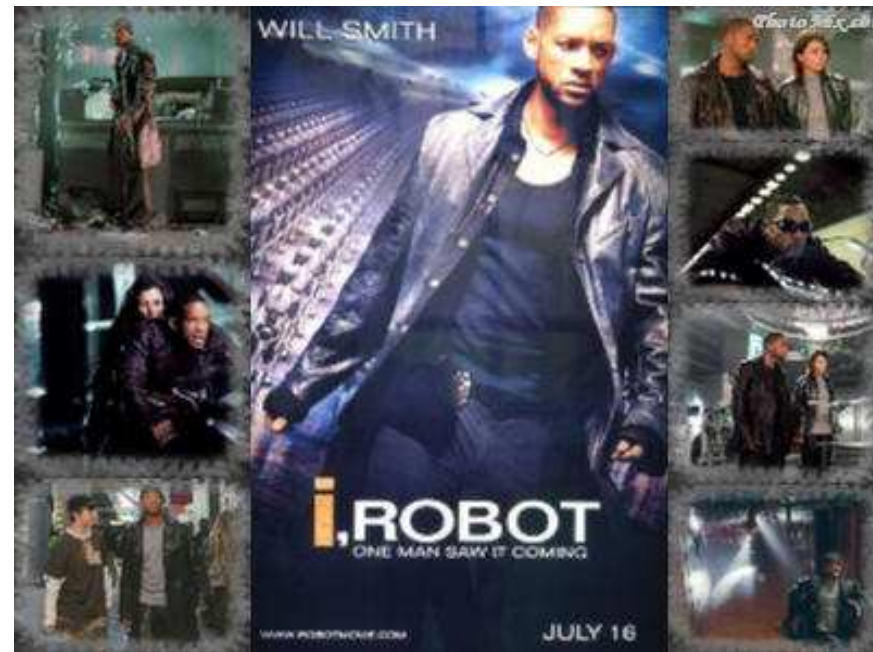
AI

Kunstmatige intelligentie (AI)

dr. Walter Kusters, Universiteit Leiden

Oegstgeest — donderdag 8 juni 2006

<http://www.liacs.nl/home/kusters/>



Kunstmatige intelligentie (AI, Artificial Intelligence) is een verzamelnaam voor een heel breed vakgebied, met onder andere:

- *robotica*: Hoe programmeer je een robot?
- *data mining*: Hoe zoekt Google?
- *rechtspraak*: Word je volautomatisch be/veroordeeld?
- *vertalen*: “the spirit is willing but the flesh is weak” → ... → “the vodka is good but the meat is rotten”?
- *computer games*: Hoe speelt de computer bij Quake?
- *neurale netwerken*: Kun je beurskoersen voorspellen?

Je kunt op minstens **twee** manieren naar Kunstmatige intelligentie kijken:

1. vanuit een meer *psychologische* of *filosofische* richting:
Wat is het verschil tussen een mens en een computer?
Kan een computer denken?
2. vanuit een meer *technische* richting:
Hoe werkt een schaakprogramma?
Hoe werkt een Marsrobot?

Kunstmatige intelligentie laat computers zich zo gedragen dat het **intelligent** zou heten als mensen het op die manier zouden doen.

De beroemde **Turing-test** (1950) stelt het volgende probleem.

In een afgesloten kamer bevindt zich een mens *of* een computer, waarmee we alleen via een computerverbinding contact kunnen hebben. Kunnen wij bepalen of het een mens is, of juist een computer?

Het originele probleem was overigens met man ↔ vrouw.

In 1964–66 schreef Joseph Weizenbaum van het MIT in Boston het beroemde programma **ELIZA/DOCTOR**, dat een psychotherapeut “speelt” — met veel succes.

Een voorbeeldconversatie:

Men are all alike

In what way

My boyfriend says I'm depressed much of the time

I am sorry to hear you are depressed

You are like my father in some ways

What resemblances do you see

Dit soort computerprogramma's doet mee in de jaarlijkse **Loebner-contest**.

Maxi en **Mini** spelen het volgende eenvoudige spel: **Maxi** wijst eerst een horizontale rij aan, en daarna kiest **Mini** een verticale kolom.

3	12	8
2	4	6
14	5	2

Bijvoorbeeld: **Maxi** kiest rij 3, daarna kiest **Mini** kolom 2; dat levert einduitslag 5.

Maxi wil graag een zo groot mogelijk getal, **Mini** juist een zo klein mogelijk getal.

Hoe spelen we dit spel zo goed mogelijk?

Als **Maxi** rij 1 kiest, kiest **Mini** kolom 1 (levert 3); als **Maxi** rij 2 kiest, kiest **Mini** kolom 1 (levert 2); als **Maxi** rij 3 kiest, kiest **Mini** kolom 3 (levert 2). Dus kiest **Maxi** rij 1!

3	12	8
2	?	?
14	5	2

Nu merken we op dat de analyse hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit. Het α - β -algoritme onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden.

Ieder schaakprogramma gebruikt deze methode.



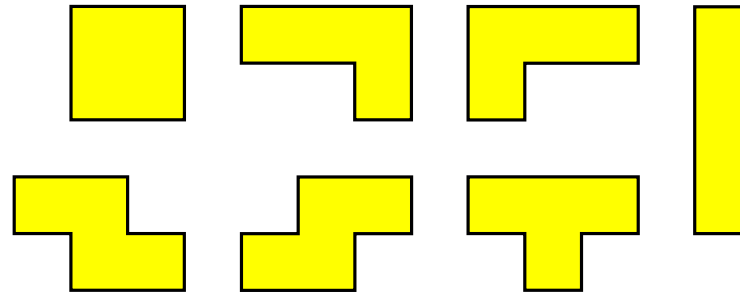
Ook aan een spel als **Tetris** kleven allerlei vragen:

- Hoe speel je het zo goed mogelijk? (AI)
- Hoe moeilijk is het? (complexiteit)
- Wat kan er allemaal gebeuren?

Zo is bijvoorbeeld bewezen dat sommige Tetris-problemen **NP-volledig** zijn, dat je bijna alle configuraties kunt bereiken, maar dat niet alle problemen “beslisbaar” zijn, zie:

<http://www.liacs.nl/home/kosters/tetris/>

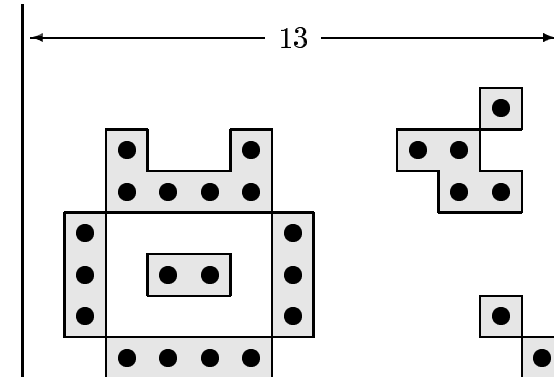
De 7 Tetris-stukken:



Stukken vallen random; volle regels worden verwijderd.
De vraag “Kun je met een gegeven serie (inclusief volgorde) van deze stukken een bord helemaal leeg spelen?” is NP-volledig.

Als iemand het bord leeg speelt kun je dat eenvoudig controleren. Als het *niet* kan, kan men (tot nu toe) niks beters verzinnen dan alle mogelijkheden één voor één na te gaan!

Een “willekeurige” configuratie:



Deze kan gemaakt worden door 276 *geschikte* Tetris-stukken op de juiste plaats te laten vallen.

Let op: alleen geheel gevulde regels verdwijnen, alles daarboven zakt *één rij*.

Claim: op een bord van oneven breedte kan elke configuratie bereikt worden!

Een **robot** is een “actieve, kunstmatige agent wiens omgeving de fysieke wereld” is. Het woord stamt uit 1921 (of eerder), en is gemaakt door de Tsjechische broers Capek. En **softbots**: RoboCom, internet programma’s.

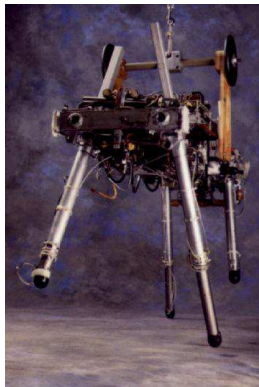
Van de science fiction schrijver Isaac Asimov (auteur van “I, Robot”) zijn de drie wetten van de **robotica**:

1. Een robot mag een mens geen kwaad doen.
2. Een robot moet menselijke orders gehoorzamen (tenzij dat tegen 1. ingaat).
3. Een robot moet zichzelf beschermen (tenzij dat tegen 1. of 2. ingaat).



NASA Sojourner

Sony Aibo



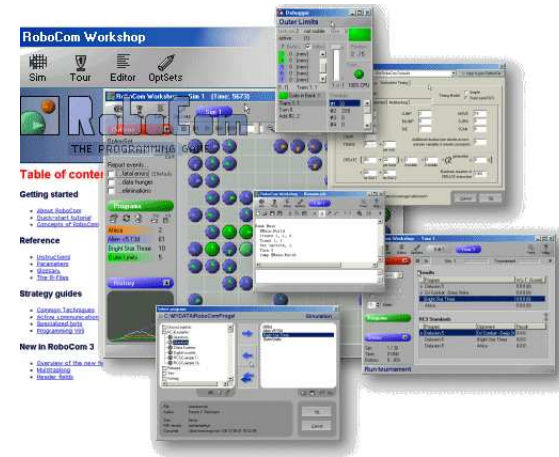
Marc Raibert's quadruped

Honda

In het programma **RoboCom** vechten robot-programma's met elkaar.

; een klein robot-programma
NAME Flooder/Shielder

```
BANK Mine           ; eerste bank
  @Loop             ; label
  TURN 0            ; draai linksom
  SCAN #5           ; scan reference field
  COMP #5,0         ; leeg?
  JUMP @Loop        ; nee, verder draaien
  CREATE 2,1,0      ; ja; creeer nieuwe robot
  TRANS 1,1         ; en kopieer jezelf
  SET %Active,1    ; activeer hem/haar
  ; auto-reboot
```



<http://www.cyty.com/robocom/>

Er zijn allerlei robot-simulaties.

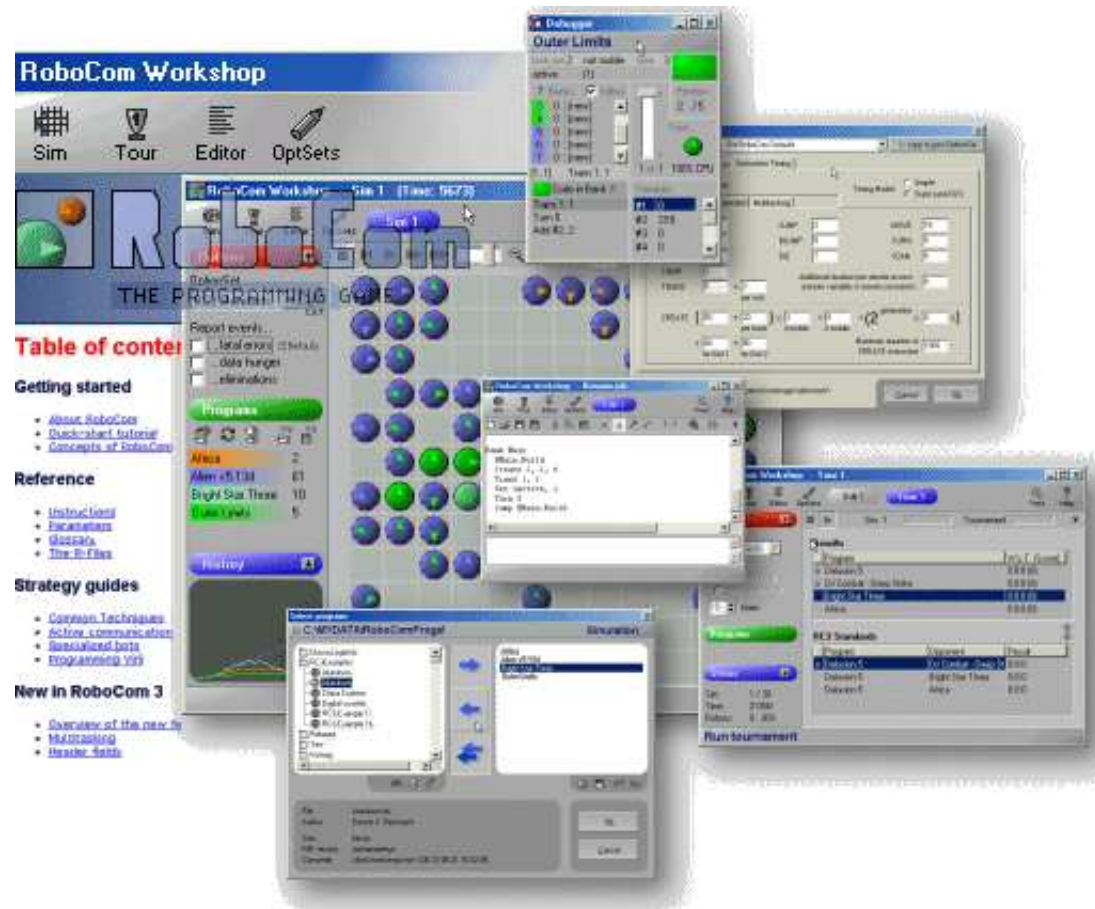
Wij bekijken **RoboCom**, een opvolger van CoreWar, gemaakt door Dennis Bemann, dat kleine robot-programma's laat vechten in een vierkant stuk computergeheugen met 20×20 vakjes (= velden).

Software (RoboCom Workshop 3.1), voor Windows XP en Linux:

`http://www.cyty.com/robocom/`

`http://www.cyty.com/robocom/download/RobSci_E.html`

De tweede link bevat alle voor ons benodigde documentatie. Er is nog meer: uitgebreide instructieset, multitasking.



Een robot is een klein assembler-achtig programma dat in één van de 20×20 velden van het “speelveld” zit. Dit veld is een “torus”: links grenst aan rechts, boven aan beneden. Een robot ziet één veld in de “kijkrichting”: het **reference field**. Een robot kan nieuwe bewegende robots maken.

Er zijn drie **instruction sets**: basic (0; met ADD, BJUMP, COMP, DIE, JUMP, MOVE, SET, SUB en TURN), advanced (1; met SCAN en TRANS erbij) en super (2; met ook nog CREATE erbij).

Een robot heeft interne integer prive-variabelen #1, #2, . . . , #20. De variabele #Active geeft aan of de robot actief is (waarde ≥ 1) of niet. En de constante \$Mobile (0/1) is de mobiliteit, \$Banks het aantal “banken” — zie verderop.

Een robot-programma is opgedeeld in maximaal 50 **banken** (\approx functies). Executie begint bij de eerste. Als je een bank uitloopt begin je weer bij de eerste bank: “auto-reboot”.

Voorbeeldprogramma, met één bank:

```
; voorbeeldprogramma, een ";" duidt op commentaar  
NAME DoetNietVeel
```

```
BANK Hoofdprogramma
```

```
  SET #3,7          ; variabele 3 wordt 7  
  @EenLabel        ; definieer een label  
  ADD #3,1         ; hoog variabele 3 met 1 op  
  TURN 1           ; draai 90 graden rechtsom (0: linksom)  
  JUMP @EenLabel   ; spring terug naar label
```

ADD #a,b	tel b bij #a op
BJUMP a,b	spring naar instructie b van bank a
COMP a,b	sla volgende instructie over als $a = b$
CREATE a,b,c	maak in reference field nieuwe robot met instruction set a, b banken en mobiliteit c
DIE	robot gaat dood
JUMP a	spring a verder (naar label @Iets mag ook)
MOVE	ga naar reference field
SCAN #a	bekijk reference field; #a wordt 0 (leeg), 1 (vijandige robot) of 2 (bevriende robot)
SET #a,b	#a krijgt waarde van b
SUB #a,b	trek b van #a af
TRANS a,b	kopieer eigen bank a naar de b-de bank van robot in reference field
TURN a	draai linksom als $a = 0$, anders rechtsom

Hierbij: #a: variabele; a, b, c: elke type.

Instructies kosten tijd, de ene meer dan de andere.

Van de eventuele robot op het reference field kun je de activiteit benaderen (en wijzigen!) via de “remote” %Active.

Als na een “auto-reboot” de eerste bank leeg blijkt, gaat de robot dood van “data-honger”.

De beginrobot staat altijd stil (mobiliteit 0).

Er zijn allerlei speciale gevallen, nog meer instructies . . .

Maak nu zelf een paar robot-programma's en vergelijk ze met elkaar en met bestaande robots.

; een klein robot-programma

NAME Mine

```
BANK Mine          ; eerste bank
  @Loop            ; label
  TURN 1           ; draai rechtsom
  SCAN #1          ; scan reference field
  COMP #1,1        ; een tegenstander?
  JUMP @Loop       ; nee, verder draaien
  SET %Active,0    ; ja, deactiveer tegenstander (eerder?)
  TRANS 2,1        ; en kopieer narigheid
  SET %Active,1    ; re-activeer
  ; auto-reboot
```

```
BANK Poison        ; tweede bank: narigheid
```

```
  DIE              ; vergif
```

```
; nog een klein robot-programma
```

```
NAME Flooder/Shielder
```

```
BANK Mine          ; eerste bank
  @Loop            ; label
  TURN 0           ; draai linksom
  SCAN #5          ; scan reference field
  COMP #5,0        ; leeg?
  JUMP @Loop       ; nee, verder draaien
  CREATE 2,1,0     ; ja; creeer nieuwe robot
  TRANS 1,1        ; en kopieer jezelf
  SET %Active,1   ; activeer hem/haar
; auto-reboot
```

1. Wat is het verschil tussen een mens en een computer?
2. Kan een computer denken?
3. Hoe bedenkt een computer een zet bij vier-op-een-rij?
4. Hoe werkt een Marsrobot?
5. Hoe werkt een vertaalprogramma?