Kunstmatige Intelligentie (AI)

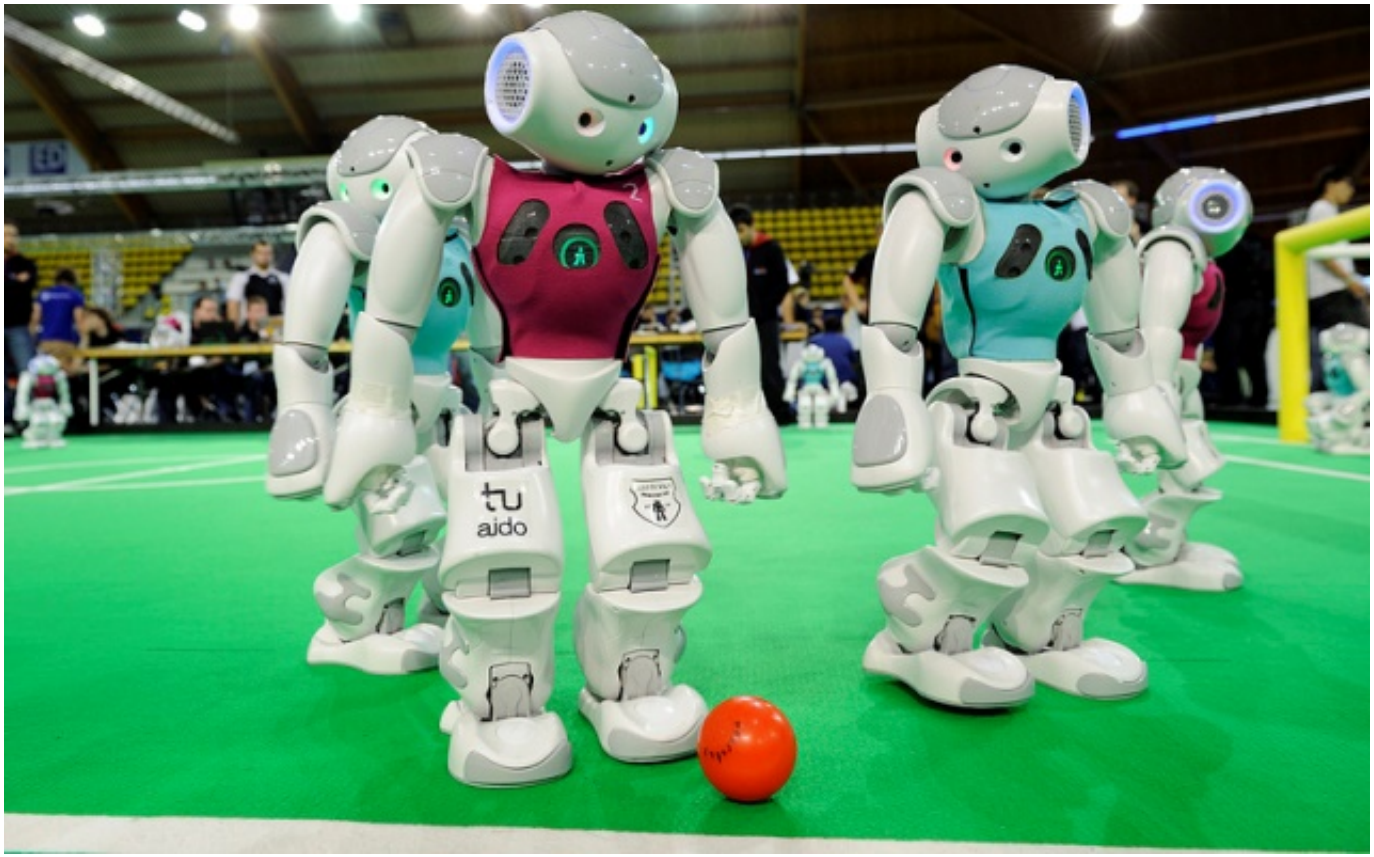
Hoofdstuk 25 van Russell/Norvig = [RN]

Robotica

voorjaar 2020

College 9, 2 april 2020

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

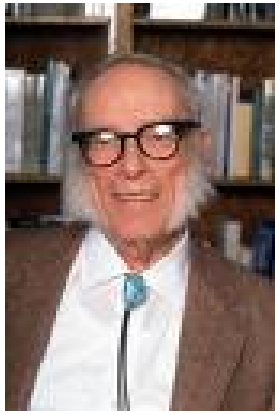


www.robocup.org

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 Čapek. En softbots: RoboCom, internet programma's.

Van de science fiction schrijver Isaac Asimov (auteur van "I, Robot") zijn de drie (later vier) 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).

Een leuke robotsimulatie, van Michael Genesereth en Nils Nilsson, is de volgende.

Bedenk een taak, bijvoorbeeld een toren maken van een paar gekleurde blokken. Stel je nu een "robot" voor die uit *vier mensen* bestaat:

Brein krijgt input van Ogen, maar kan zelf niet zien; geeft opdrachten aan Handen; maakt plannen

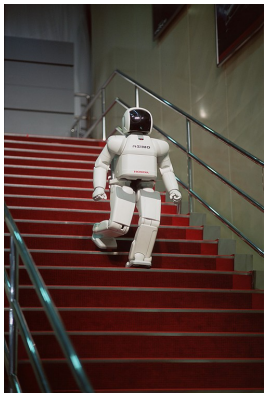
Ogen weet niet wat het doel is

Handen, Links en Rechts voeren simpele opdrachten uit; ze zijn geblinddoekt

Mobiele robots

Curiosity

Sony Aibo



Hiroshi Ishiguro's robot (en hijzelf)
"uncanny valley"

Honda ASIMO

Enkele begrippen uit de robot-wereld zijn:

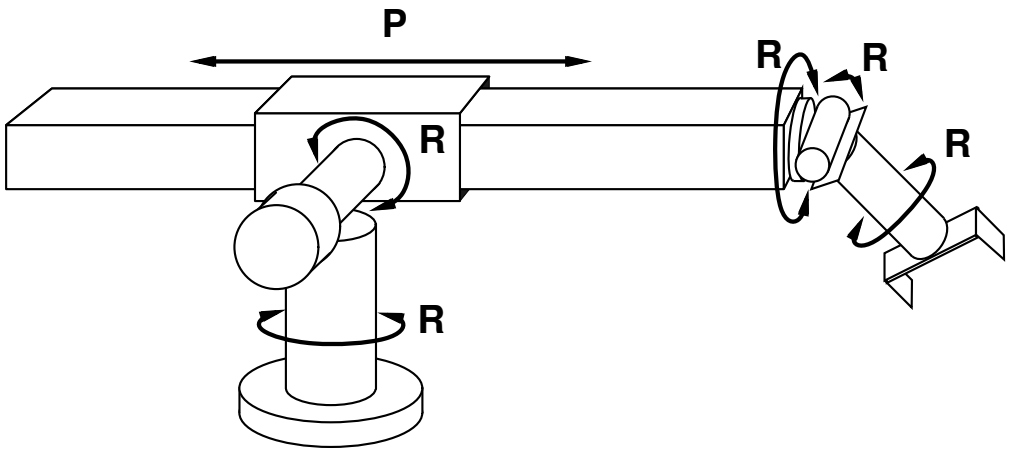
statisch stabiel stabiel zonder te bewegen

dynamisch stabiel robot verongelukt bij gedwongen rust

holonome robot totaal aantal vrijheidsgraden (onafhankelijke richtingen waarin robot of zijn "effectors" kunnen bewegen) = aantal controleerbare vrijheidsgraden
Voorbeeld niet-holonoom: auto (effectief 3: x, y, θ resp. 2: vooruit/achteruit, draaien); erger: met aanhanger

configuratie-ruimte bestaande uit bijvoorbeeld 6 hoeken voor een robot-arm

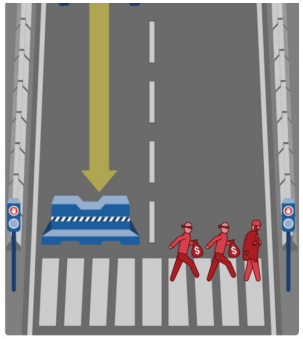
De Stanford manipulator ziet er als volgt uit:

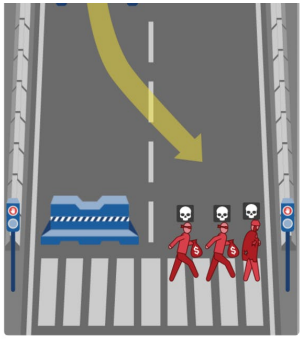


P: prismatic joint; R: revolute joint
De configuratie van de robot wordt gespecificeerd door 6 getallen: dus 6 vrijheidsgraden ("degrees of freedom" = DOFs).

De arm van de mens heeft $>$ 6 vrijheidsgraden (plaats hand op tafel en beweeg elleboog).

Momenteel doen zelfrijdende systemen het erg goed.





$^{\uparrow}$computer-zien, kennis, de wet, ethiek(†)$^{\uparrow}$, . . .

Udacity en Coursera $\longrightarrow$ MOOC's over zelfrijdende auto's
(†) zie Michael Sandel's video's op YouTube

Een robot moet —in een 2-dimensionale wereld— van begin naar doel, weet precies waar hij is (GPS), en heeft een tastsensor. Een online-algoritme voor zo'n robot is:

1. Bereken de rechte lijn ℓ van begin naar doel, en begin langs ℓ richting doel te bewegen.

2. Als je botst: onthoud positie Q. Wandel met de klok mee om het obstakel heen, terug naar Q. Detecteer onderweg waar je ℓ kruist, en onthoud welk punt P het dichtst bij het doel is.

3. Loop van Q naar P (met de klok mee of er tegenin, wat het kortste is), en ga via 2. zo verder.

Dit algoritme heeft een "competitive ratio" van ten hoogste $1 + 1.5B/|\ell|$ ($|\ell|$: afstand van begin naar doel via ℓ; B: som van lengtes van randen van obstakels), de slechtst mogelijke verhouding tussen gevonden en kortste route.

Bewijs: de verhouding gevonden staat tot kortste is hooguit

$$\frac{|\ell| + 1.5B}{|\ell|} = 1 + 1.5B/|\ell|,$$

en je kunt die grens bereiken: leg een rafelig smal object met omtrek B op (en in de richting van) ℓ; de kortste weg heeft dan lengte $\approx |\ell|$.

De subsumption architectuur is een idee van Rodney Brooks van MIT uit 1986, zie zijn artikel (via de AI-website).

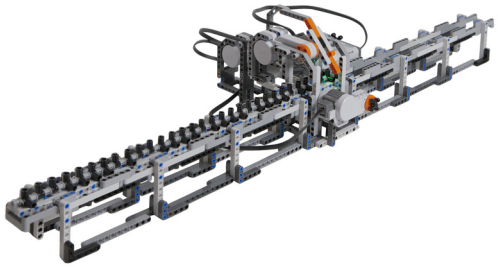
Een robot bestaat hier uit verschillende *finite state machines* met klokken. Deze worden netjes gesynchroniseerd en gecombineerd. Zo vermijd je problemen met gigantische configuratieruimtes. Ze worden ook gebruikt om NPC's ("non-playing characters") in computerspellen te modelleren.

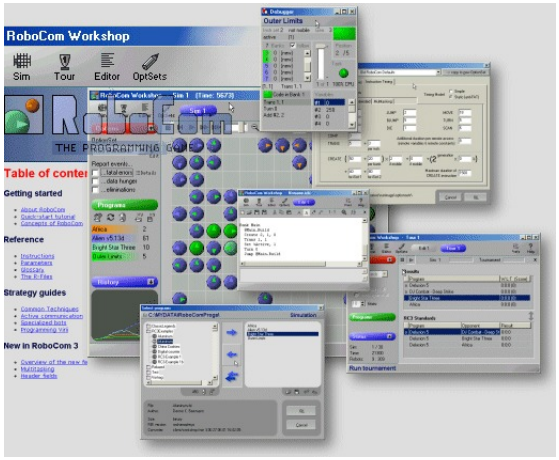
Het werkt goed voor één kleine taak, maar wat er gebeurt is soms lastig te snappen. Zie je hiervan iets terug bij Lego/RoboCom?

Twee voorbeelden

- hardware: Lego

- software: RoboCom (zie vroeger)

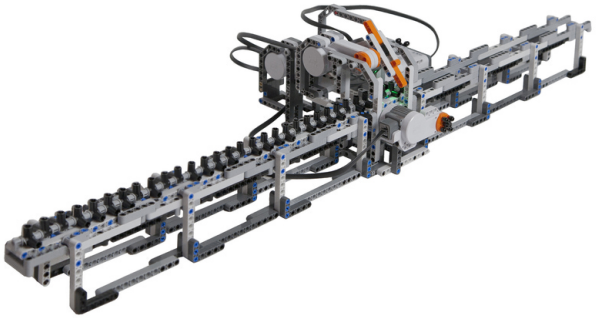




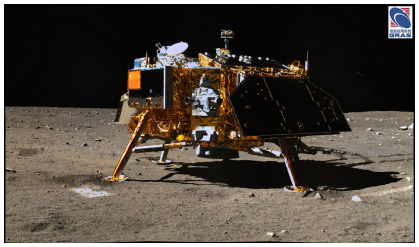
Lego-robots — en meer ☰

A simpel programma voor een Lego-robot laat deze "random" lopen — en dat ziet er intelligent uit.

```
task main ( ) {
  while ( true ) {
    OnFwd (OUT_A + OUT_C);
    Wait (Random (100) + 40);
    OnRev (OUT_A);
    Wait (Random (85) + 30);
  }
}
```



Lego Turing-machine



Chang'e-3 maanlander

Pepper-robot van Softbank (2016)

Je kunt Lego-robots programmeren in Dave Baum's NQC (Not Quite C) met behulp van Mark Overmars' RCX Command Center, zie handleidingen op de websites.

```
#define MOVE_TIME 100
#define TURN_TIME 85
task main ( ) {
  while ( true ) {
    OnFwd (OUT_A + OUT_C);  // motoren op A en B aan
    Wait (Random (MOVE_TIME) + 40);
    OnRev (OUT_A);          // en die op A achteruit
    Wait (Random (TURN_TIME) + 30);
  }//while
}//main
```

Nadelen: maximaal 32 (integer) variabelen, maximaal 10 "parallelle" tasks, alleen eenvoudige functies: subroutines (`sub`), online functies (`void`) en macro's (`#define`).


mindstorms


EV3

Er zijn tastsensoren en lichtsensoren, met 3 aansluitpunten:

```
task main ( ) {
  SetSensor (SENSOR_1,SENSOR_TOUCH);  // 0 of 1
  SetSensor (SENSOR_2,SENSOR_LIGHT);  // 0..100
  SetSensor (SENSOR_3,SENSOR_TOUCH);
  OnFwd (OUT_A + OUT_C);
  while ( true ) {
    if ( SENSOR_1 == 1 ) ...
    else if ( SENSOR_2 > 40 ) ...
  }//while
}//main
```

Er kunnen ook meerdere sensoren op één input worden gezet, en ze kunnen preciezer (“raw”) worden uitgelezen.

```
task main ( ) {
  SetSensor (SENSOR_1,SENSOR_TOUCH);
  start check_sensors;
  while ( true ) {
    OnFwd (OUT_A + OUT_C); Wait (100);
    OnRev (OUT_C); Wait (85);
  }//while
}//main
task check_sensors ( ) {
  while ( true ) {
    if ( SENSOR_1 == 1 ) {
      OnRev (OUT_A + OUT_C); Wait (50);
      OnFwd (OUT_A); Wait (85); OnFwd (OUT_C);
    }//if
  }//while
}//check_sensors
```

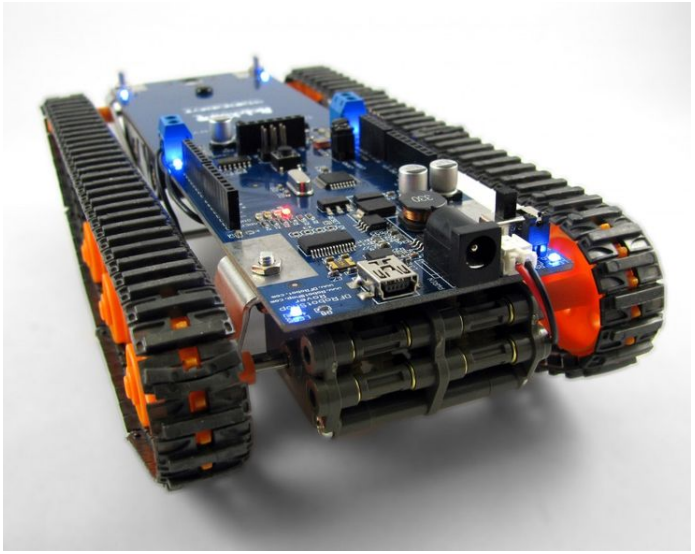
Deze acties interfereren! Oplossing: of `stop` of semaforen:

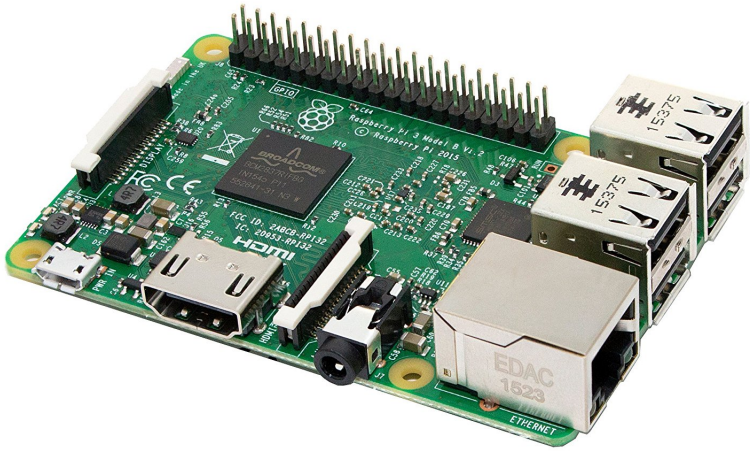
```
until ( sem == 0 ); sem = 1; /*motoractie*/ sem = 0;
```

```
int level;
task geefsignaal ( ) {
  while ( true ) { SendMessage (0); Wait (10); }//while
}//geefsignaal
task checksignaal ( ) {
  while ( true ) {
    level = SENSOR_2;
    if ( SENSOR_2 > level + 200 ) {  // snelle fluctuatie
      OnRev (OUT_C); Wait (85); OnFwd (OUT_A+OUT_C);
    }//if
  }//while
}//checksignaal
task main ( ) {
  SetSensor (SENSOR_2,SENSOR_TYPE_LIGHT);
  SetSensorMode (SENSOR_2,SENSOR_MODE_RAW);  // 0..1023
  OnFwd (OUT_A+OUT_C);
  start geefsignaal;
  start checksignaal;
}//main
```

In werkelijkheid zit de “ruwe” waarde tussen 300 (licht) en 800 (donker).

Arduino, www.arduino.cc





Raspberry Pi, www.raspberrypi.org

Het huiswerk voor de volgende keer (donderdag 9 april 2020): lees Hoofdstuk 18.7, p. 727–737 van [RN], over Neurale netwerken door. Kijk naar de vragen bij dit hoofdstuk.

Werk aan de derde programmeeropgave: Othello; kijk naar de video's. Deadline: 23 april 2020.

NB: Ξ betekent "geen tentamenstof".