**En situerad ansats för utvecklingen av en räknande robot**

**(HS-IDA-EA-03-501)**

**Niclas Ahlén (a00nicah@student.his.se)**

*Institutionen för datavetenskap*
*Högskolan i Skövde, Box 408*
*S-54128 Skövde, SWEDEN*

Examensarbete på det kognitionsvetenskapliga programmet under vårterminen 2003.

Handledare: Mikael Thieme

**En situerad ansats för utvecklingen av en räknande robot**

Examensrapport inlämnad av Niclas Ahlén till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för Datavetenskap.

**2003-06-06**

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

**En situerad ansats för utvecklingen av en räknande robot**

**Niclas Ahlén (a00nicah@student.his.se)**

# Sammanfattning

Den situerade ansatsen inom artificiell intelligens har i tidigare experiment visat på stora möjligheter vid utvecklingen av enkla beteenden. Ansatsens framgångar är dock inte lika tydliga när det kommer till utvecklingen av mer komplexa beteenden som i högre grad påminner om de experiment som gjorts inom traditionell artificiell intelligens. I studien utvecklas en agent med ett ”Extended Sequential Cascaded Network” som kontrollarkitektur för att lösa en uppgift som kräver ett ”räkneliknande beteende”. Utvecklingen av nätverket grundas på en situerad syn på kognition, däribland att designern i så liten grad som möjligt skall styra utvecklingen. Experimentets resultat visar på en agent som inte ens löser den enklaste versionen av uppgiften. I diskussionen härleds misslyckandet till svårigheterna med en designeroberoende utveckling.

**Nyckelord:** kognitionsvetenskap, extended sequential cascaded network, situerad ansats, interna tillstånd, räkneliknande beteende

# Innehållsförteckning


**1 Introduktion ..... 1**

**2 Bakgrund ..... 3**

2.1 Traditionella representationer ..... 4

2.1.1 Traditionell AI ..... 4

2.1.2 Konnektionism ..... 5

2.2 Interaktionsbaserad representation ..... 6

2.2.1 Radikal konnektionism ..... 7

2.3 Dynamiska system ..... 7

2.4 Situerad AI ..... 9

2.5 Inlärning för autonom robot ..... 10

2.6 Fitnessfunktion ..... 11

2.7 Simulering ..... 12

2.8 Tidigare arbeten ..... 12

2.8.1 Kontrollarkitektur för räkning ..... 12

2.8.2 Kontrollarkitektur för fördröjt svar ..... 14

**3 Problemprecisering ..... 17**

3.1 Avgränsning ..... 17

**4 Experimentet ..... 18**

4.1 Experimentets upplägg ..... 18

4.1.1 Agentens uppgift ..... 18

4.1.2 Yet Another Khepera Simulator (YAKS) ..... 18

4.1.3 Agentens kontrollarkitekturer ..... 20

4.1.4 Implementation ..... 20

4.1.5 Validering av implementation ..... 22

4.2 Utvärdering av experimentet ..... 22

4.3 Kompletterande experiment ..... 23

**5 Resultat och analys ..... 25**

5.1 Agentens utveckling ..... 25

5.2 Pålitlighet ..... 26

5.3 Agentens hantering av uppgiften ..... 27

5.3.1 ESCN ..... 27

5.3.2 Enkelt återkopplat nätverk ..... 31

**6 Diskussion ..... 34**
6.1 Fortsatta studier ..... 35
**Referenslista ..... 36**

# 1 Introduktion

Tidigare har artificiell intelligens (AI) handlat om att modellera högre kognitiva förmågor såsom schackspelande, missionär och kannibal-problem samt annan problemlösning. Modellerna har anpassats för den specifika domänen, vilket på grund av deras intellektuella nivå har lett till abstrakt resonerande system som hanterar symboler. Systemen har inte behövt interagera direkt med omgivningen, utan forskarna har varit länken mellan modell och verklighet, och förberett input samt tolkat output (Brooks, 1991a).

Idén med förberedd input vilar på synen av en objektiv verklighet där varje agent uppfattar omgivningen på samma sätt oberoende behov, bakgrund, kontext etcetera. Detta synsätt förtydligas med Palmers (1978) beskrivning av traditionell representation, vilken har en tydlig koppling till traditionell AI. Traditionell representation innebär att ett externt objekt motsvaras av en intern representation hos agenten. Länge har den här typen av representation stått oemotsagd inom kognitionsvetenskapen och ofta har likhetstecken dragits mellan kognition och manipulering av traditionella representationer (van Gelder, 1999).

Enligt Beer (1995) ger språket oss metaforer och koncept som sätter gränserna för våra resonemang. Används begrepp såsom representation leder det till en fokusering kring dess struktur, var de finns lagrade, hur de återhämtas, vad de betyder och så vidare. Beer (2000) menar att denna ansats syn på kognition har lett till klena modeller av isolerade kognitiva fenomen. Modellerna har inte givit någon djup förståelse av kognition, tvärtemot vad ansatsens förespråkare utlovat. Det är därför viktigt att ifrågasätta rollen av traditionell representation inom kognitionsvetenskapen. Med nya ansatser kommer fokus att förändras och förhoppningsvis ge djupare förståelse inom området.

Sedan mitten av 1980-talet har en situerad syn på kognition fått större plats på den kognitionsvetenskapliga scenen (Ziemke, 2000). Med idén att kognition är en del av tänkande, kropp och omgivning har fokus förflyttats från en abstrakt symbolmanipulation till fysiska robotar som interagerar med omgivningen. Det som tidigare har ansetts varit det enda möjliga ansatsen gentemot kognition kan idag alltså ifrågasättas.

Beer (1995) är intresserad av att få veta möjligheterna med den nya ansatsen. Han menar att den i flera experiment har fungerat väl på enklare uppgifter, men undrar om den verkligen räcker till för sånt som kräver högre kognitiva förmågor.

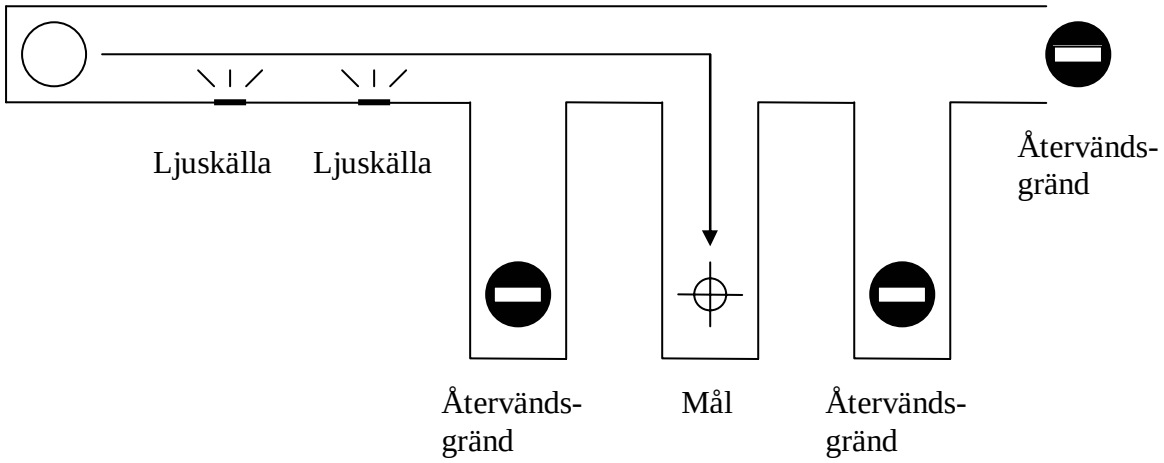
En uppgift som kan anses vara en högre kognitiv förmåga och som intuitivt kräver traditionell representation är räkning. Tidigare experiment (Wiles & Elman, 1995) har dock visat på möjligheten för ett artificiellt neuralt nätverk som saknar vissa av de egenskaper som finns hos en räknare att lösa en räknekrävande uppgift. Experimentet var dock osituerat, vilket minskar möjligheterna att generalisera beteendet till en fysisk agent.

En arkitektur som tidigare använts i situerade experiment är ”Extended Sequential Cascaded Network” (ESCN)(Thieme, 2002; Ziemke, 1999, 2000; Ziemke & Thieme, under tryckning). Denna kontrollarkitektur har visats klara av uppgifter som kräver korttidsminne utan att använda sig av traditionell representation.

I rapporten beskrivs ett situerat experiment med kontrollarkitekturerna ESCN respektive enkelt återkopplat nätverk. Agentens uppgift är att lösa ett problem som

antas kräva ett räkneliknande beteende, det vill säga ett beteende som beror på ett antal ljusstimuli som presenterats för agenten. Stimuli presenteras skilt från den del av uppgiften som fordrar agentens respons, vilket innebär att agenten måste komma ihåg nödvändig information om stimulit under en viss tidsperiod (den måste ha ett korttidsminne).

Agentens uppgift är att välja gång *n* givet att den passerat *n* antal lampor tidigare i korridoren, se Figur 1. Givet 1 lampa skall den svänga in i den första gången, givet 2 lampor skall den svänga in i den andra gången och så vidare.




**Figur 1: Uppgift som kräver att agenten har ett räkneliknande beteende för att lösas.**

Målet med arbetet är att undersöka de interna tillstånden hos en agent med kontrollarkitekturen ESCN när den ställs inför en uppgift som kräver ett räkneliknande beteende.

# 2 Bakgrund

Tre olika synsätt på interna tillstånd kommer att presenteras. I avsnitt 2.1 beskrivs Palmers (1978) syn på interna tillstånd. Enligt denna ansats fungerar de interna tillstånden som representationer vilka speglar den externa verkligheten. Ett objekt i omvärlden representeras av ett visst internt tillstånd i den inre världen (Dorffner, 1997). I avsnittet beskrivs två ansatser inom kognition som enligt Dorffner (1997) anammat denna syn: traditionell AI och (icke-radikal) konnektionism.

I avsnitt 2.2 presenteras Dorffners (1997) syn på interna tillstånd som interaktionsbaserade representationer. Med interaktionsbaserade representationer ses interna tillstånd fortfarande som representationer, men de kan enbart förstås utifrån agentens drifter och behov. Ett internt tillstånd kan representera en situation trots att det inte finns någon specifik struktur i omgivningen som givit upphov till det interna tillståndet. Tillståndet kan uppkomma genom flera olika stimuli tillsammans eller genom agentens förväntningar på omvärlden. I avsnittet beskrivs radikal konnektionism (Dorffner, 1997) som en ansats direkt sprungen ur dessa idéer.

I avsnitt 2.3 beskrivs hur interna tillstånd kan ses som en del av dynamiska system (Beer, 1995, 2000; van Gelder, 1999). Systemets tidigare erfarenhet leder till att det befinner sig i ett visst internt tillstånd. Det tidigare interna tillståndet påverkar vilket nytt internt tillstånd som systemet hamnar i givet en viss input. Input som vid ett tillfälle givit ett visst internt tillstånd, kan i nästa stund leda till något helt annat internt tillstånd. Det finns alltså inte någon överensstämmelse mellan ett visst internt tillstånd och en representation av ett externt tillstånd (Beer, 2000).

Enligt Beer (2000) blir den dynamiska ansatsen som mest kraftfull tillsammans med situerad AI. Denna beskrivs i avsnitt 2.4. Idén om att systemet skall vara situerat lyfts även fram av Brooks (1991a).

Vidare beskrivs tre olika typer av inlärning. I avsnitt 2.5 beskrivs oövervakad inlärning (eng. unsupervised learning), övervakad inlärning (eng. supervised learning), samt inlärning genom belöning (eng. reinforcement learning). Här beskrivs även Meedens (1996) uppdelning av lokal respektive global metod för inlärning genom belöning.

Resultatet av inlärning genom belöning beror till stor del på fitnessfunktionens utformning. Nolfi och Floreano (2000) föreslår fitnessrymden som ett objektivt mått för att beskriva fitnessfunktionen. I avsnitt 2.6 beskrivs dimensionerna i fitnessrymden samt hur en fitnessfunktion för en autonom robot som agerar i oförutsägbar miljö bör se ut.

Om agentens inlärning sker i en simulator ger detta möjligheten till en betydligt snabbare utveckling än om den sker i en verklig miljö. I avsnitt 2.7 beskrivs Jakobis (1997) uppdelning av den simulerade miljön i basmängd och implementationsaspekter.

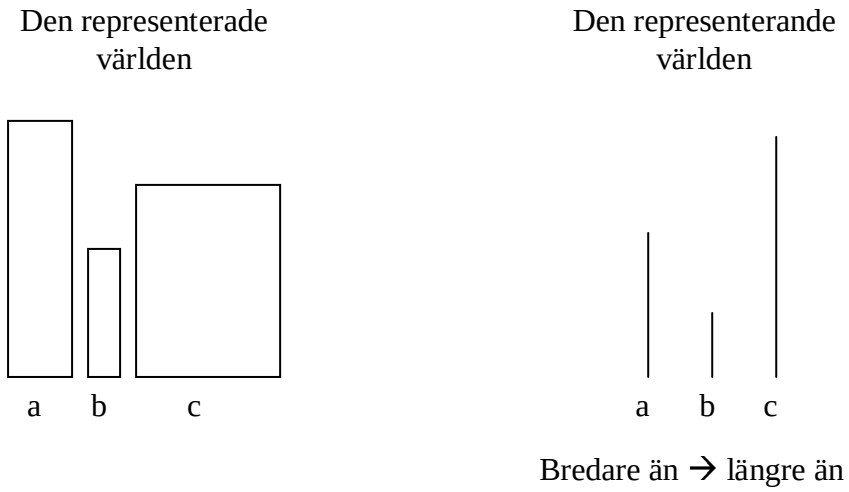
I den avslutande delen, avsnitt 2.8, tas det upp tidigare arbeten inom området. Först beskrivs ett experiment där ett enkelt återkopplat nätverk lär sig att räkna (Wiles & Elman, 1995). Vidare beskrivs ett experiment där en kontrollarkitektur med virtuell modularitet används för att lösa ett problem som kräver korttidsminne (Ziemke & Thieme, under tryckning).

## 2.1 Traditionella representationer

Enligt Palmer (1978) är en representation något som står för någonting annat. Den är en modell som belyser några av de egenskaper hos den eller de saker som representeras. Detta antagande utgår från att det finns två världar: den representerande världen och den representerade världen. Palmer (1978) menar att en representation egentligen är ett representerande system bestående av fem delar:

1. vad den representerade världen är
2. vad den representerande världen är
3. vilka aspekter hos den representerade världen som modelleras
4. vilka aspekter hos den representerande världen som modellerar
5. överensstämmelsen mellan de två världarna

I Figur 2 visas ett exempel på de fem delarna i en representation.




**Figur 2: Exempel på en representation. Relationen ”bredare än” i den representerade världen modelleras genom linjernas längd i den representerande världen. Enhet a i den representerade världen motsvaras av enhet a i den representerande världen (efter Palmer, 1978, s.263).**

Enligt Ziemke (2000) har denna förklaring av representationer en tydlig koppling till traditionell AI genom tolkningen av en inre representation som motsvarighet till ett externt objekt.

### 2.1.1 Traditionell AI

Inom traditionell AI antas inre representationer referera till externa objekt och kognition anses vara hantering av de inre representationerna. Newell och Simons (1997) fysiska symbolsystemhypotes är ett exempel på den traditionella AI:ns syn på kognition.

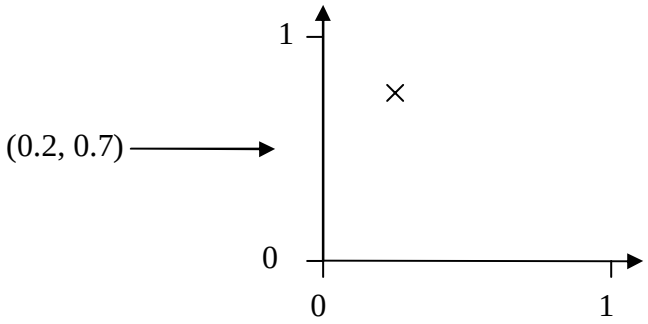
Newell och Simon (1997) beskriver ett fysiskt symbolsystem som en mängd symboler vilka tillsammans kan bilda uttryck. Ett uttryck refererar till ett objekt om systemet antingen kan påverka objektet eller bete sig på ett sätt som beror på objektet. Utöver symboler och uttryck består systemet av en samling processer som kan skapa nya uttryck från de uttryck som redan finns. Enligt Newell och Simon (1997) har ett fysiska symbolsystem stor betydelse för vår förståelse av intelligens.

Enligt Beer (2000) utgörs en typisk modell inom traditionell AI av ett datorprogram som får en beskrivning av ett problem i form av symboliska representationer. Programmet löser sedan problemet med hjälp av den kunskap som redan finns om problemdomänen i systemet. Problembeskrivningen manipuleras rent syntaktiskt för att nå en lösning på problemet. Beer (2000) menar att förklaringsfokus inom ansatsen ligger på representationernas struktur och innehåll samt på de algoritmer som används för manipulering.

### 2.1.2 Konnektionism[1]

En alternativ ansats som använder sig av traditionell representation är konnektionismen. Denna ansats skiljer från traditionell AI genom vad den representerande världen är, se punkt 2 ovan (Niklasson & Bodén, 1997).

De konnektionistiska modellerna består av nätverk med flera lager av enkla neuronliknande enheter. Nätverken tränas att omvandla numerisk input till numerisk output (Beer, 2000). Input och output består av reella vektorer där en vektor motsvarar en representation av ett objekt (Dorffner, 1997). Den representerande världen har formen av en multidimensionell rymd där representationerna utgör punkter i rymden, se Figur 3.




**Figur 3: En 2-dimensionell vektor och dess motsvarighet i en 2-dimensionell rymd.**

Enligt Niklasson och Bodén (1997) ligger vanligen uttryck som är lika nära varandra och uttryck som är olika långt ifrån varandra. Det spatiala avståndet mellan två uttryck är alltså relevant, vilket är en egenskap som inte finns hos modeller inom traditionell AI. Detta är något som avspeglas i hur systemen manipulerar uttrycken: ett fysiskt symbolsystem förändrar uttryckens syntaktiska form och konnektionistiska system förändrar uttryckens spatiala form (Niklasson & Bodén, 1997).

Kritik gentemot att traditionell AI och konnektionism skall leda till intelligenta system har bland annat kommit från Searle (1997), med argumenten om det kinesiska rummet och mongolen med vattenrören. Searle (1997) menar att systemen inte har någon förståelse eller några intentioner med sitt handlande, utan att de enbart sysslar med meningslös manipulering av formella strukturer (symboler i traditionell AI och vektorer inom konnektionism) som har betydelse för designern men inte för systemet självt. Även Dreyfus (1997) kritiserar den traditionella AI:n. Han kritiserar den för att skapa system där all kunskap behöver finnas explicit representerat. Enligt Dreyfus (1997) finns det aspekter, exempelvis våra känslor, som påverkar hur vi uppfattar

[1] I arbetet kommer inte grunderna inom konnektionistiska nätverk att tas upp. För mer information om konnektionistiska nätverk se Mehrotra, Mohan och Ranka (1997).

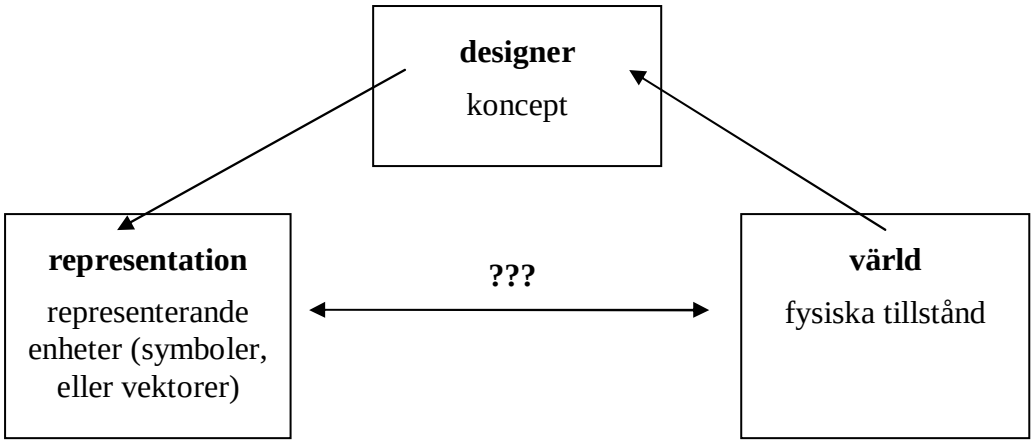
omgivningen och dessa kan inte fångas med representationer. Han påstår vidare att allt det som har lett fram till den situation agenten befinner sig i är alldeles för omfattande för att kunna representeras på traditionellt vis, det vill säga med explicita fakta och regler. Dreyfus (1997) menar att agenten behöver vara situerad (se 2.4) för att kunna uppnå intelligens.

För att komma över problemen med traditionell AI och tidig konnektionism skapades den radikala konnektionismen (se 2.2.1). I denna ansats används en annan typ av representationer än inom traditionell AI och konnektionism (Dorffner, 1997).

## 2.2 Interaktionsbaserad representation

Enligt Dorffner (1997) använder den traditionella AI:n modeller som inte speglar verkligheten, utan snarare speglar designerns syn på verkligheten, se Figur 4. Han baserar detta på antagandet om att det inte finns någon objektiv värld, utan enbart subjektiva (agentberoende) bilder av omvärlden. Konnektionismen faller in under samma kritik. Dorffner (1997) menar att designern ger det konnektionistiska nätverket förberedda representationer som input och använder utvalda träningsexempel för att träna nätverket. Det lämnas lite kvar av uppgiften och den blir väldigt enkel för nätverket att lösa. Även Brooks (1991b) tar upp problemet med förberedda representationer:

> ”…this abstraction is the essence of intelligence and the hard part of the problems being solved. Under the current scheme the abstraction is done by the researchers leaving little for the AI programs to do but search.” (Brooks, 1991b, s.141)




**Figur 4: Representationernas koppling till världen, inom traditionell AI och konnektionism (efter Dorffner, 1997, s.101).**

Dorffner (1997) beskriver en typ av representation som kallas för interaktionsbaserad representation (eng. interactivist representation). Begreppet syftar till de interna strukturerna som finns för att *vägleda agentens beteende*. En sådan struktur behöver inte överensstämma med någon specifik extern struktur, utan kan uppkomma från flera olika stimuli tillsammans eller genom agentens egna förväntningar på omgivningen. Betydelsen av en interaktionsbaserad representation kan enbart bestämmas utifrån agenten själv, dess drifter och behov (Dorffner, 1997). Detta skiljer sig alltså från Palmers (1978) syn på representation som *spegelbild av omvärlden*.

Interaktionsbaserade representationer är alltså inte objektiva representationer och kan inte tolkas utan förståelse för agentens förutsättningar.

En ansats som är tydligt kopplad till interaktionsbaserade representationer är radikal konnektionism. Dorffner (1997) beskriver den radikala konnektionismen som att vara fri från den agentoberoende synen på representation, det vill säga fri från den syn där designern bestämmer instantieringar av representationer.

### 2.2.1 Radikal konnektionism

Enligt den radikala konnektionismen (Dorffner, 1997) bör de interna representationerna skapas genom självorganisering (eng. self-organization). Självorganisering innebär att agenten automatiskt anpassar sitt beteende och de interna representationerna i interaktion med omvärlden, till exempel genom inlärning eller evolution.

Designern påverkar fortfarande utvecklingen av systemen, men på en mer generell nivå i jämförelse med traditionell AI och konnektionism. Inom radikal konnektionism är det inte tillåtet för designerna att bestämma specifika instantieringar av representationer, såsom att ett A betyder stol eller att (0.2, 0.5, 1.2) betyder bord. Designern måste däremot bestämma strukturerna inom vilka representationerna kan uppkomma, exempelvis att systemet kan kategorisera och ha relationer mellan olika kategorier (Dorffner, 1997).

Agenten måste alltså befinna sig i omgivningen och lära sig från denna genom sina egna kroppsliga erfarenheter. Representationerna skapas således enbart genom agentens användande av input från sensorerna och output till motorerna. Detta leder enligt Dorffner (1997) till att alla representationer grundas i agentens input och output. Även högre nivåerna av kognition, såsom koncept och kategorier, måste grundas i dessa sensomotoriska erfarenheter.

Denna syn på kognition har många likheter med en situerad ansats till AI (se avsnitt 2.4).

## 2.3 Dynamiska system

Dynamiska system är ännu ett bra exempel på hur kognition kan modelleras utan representationer i Palmers (1978) mening. Inom dynamiska system används modeller som uttrycker systemets förändring över tid. Agentens tidigare erfarenhet påverkar den kommande interaktionen med omgivningen genom att det interna tillståndet hela tiden förändras av inputen. Input som tidigare har givit ett visst internt tillstånd, kan i nästa stund vara källa till något helt annat internt tillstånd. Det behöver alltså inte finnas någon överensstämmelse mellan ett visst internt tillstånd och en representation av ett externt tillstånd (Beer, 2000). Representationer i Palmers (1978) mening är inte något som helt och hållet bestrids av ansatsen, men däremot tillbakavisas den traditionella AI:ns och konnektionismens antagande om att all kognition kan förklaras med hjälp av denna typ av representationer (van Gelder, 1999).

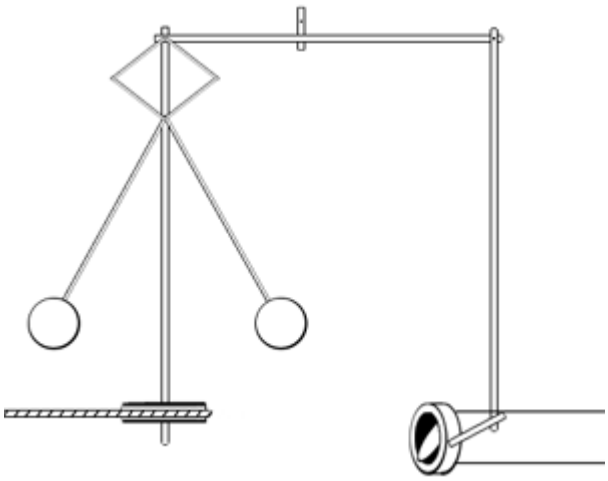
Enligt Beer (1995) kan en agents beteende inte förklaras enbart utifrån agenten själv, utan snarare från dess interaktion med omgivningen. Tillsammans kan flera system som påverkar varandra skapa betydligt mer komplexa beteenden än vad de kan göra var för sig. Egenskaperna eller beteendena som uppkommer kan då inte tillskrivas ett visst system ensamt. Beer (1995) menar därför att det skall ses som att en agent har möjligheten till ett visst beteende, men att det bara är när agenten befinner sig i en passande miljö som den faktiskt kan utföra ett beteende.

Interaktionen mellan agenten och omgivningen skall alltså inte ses som tillfällig input och output, utan snarare som en ömsesidig, kontinuerlig anpassning. Van Gelder (1999) jämför den traditionella AI:ns ansats med den dynamiska ansatsen med hjälp av hur de modellerar lösningen på ett problem. Problemet handlar om hur ett tillbehör skall designas för att anpassa en ångmaskin så att den kör på en konstant hastighet trots alla de faktorer som orsakar variation i hastigheten. Den traditionella AI:n hade löst det med den här algoritmen (van Gelder, 1999):

1. mät ångmaskinens hastighet
2. jämför ångmaskinens hastighet med önskad hastighet
3. om det inte är någon skillnad, gå till steg 1; annars
    a. mät ångtrycket
    b. räkna ut önskad förändring i trycket
    c. räkna ut förändringen i spjällets läge
4. utför förändringen
5. gå till steg 1

Enligt denna lösning krävs det först att systemet får måtten (hastighet, önskad hastighet, ångtryck etcetera) som input vilket resulterar i representationer av de relevanta aspekterna hos systemet. Dessa representationer förändras sedan enligt vissa regler och görs slutligen om till en form som passar för att utföra förändringen av spjällets läge (van Gelder, 1999).

En lösning på problemet som ligger i enlighet med den dynamiska ansatsen är Watt governor, se Figur 5.



**Figur 5: Watt governor (efter Bechtel, 1997).**

Watt governor består av en vertikal stång som sitter ihop med ångmaskinen. Stångens rotationshastighet är direkt beroende av ångmaskinens hastighet. Överst på stången sitter två metallarmar med varsin metallkula ytterst. När stången roterar lyfts metallarmarna upp på grund av centrifugalkraften. Ju högre rotationshastighet metallarmarna har, desto högre upp lyfts armarna. Armarnas höjd påverkar i sin tur spjällets läge. Höjs armarna förändras spjällets läge så att mindre ånga kommer ut, vilket leder till att ångmaskinens hastighet sänks. Om öppningen för ångan är för liten

får ångmaskinen en låg hastighet. Den låga hastigheten gör att stången roterar saktare och metallarmarna sjunker. Sänks metallarmarna förändras spjällets läge så att mer ånga kommer ut, vilket leder till att ångmaskinens hastighet ökar (van Gelder, 1999). Lösningen kan beskrivas av en differentialekvation som specificerar systemets dynamik och kan implementeras på en dator.

Den dynamiska lösningen skiljer sig på flera sätt från den traditionella AI:ns lösning. Istället för stundvis input, symboliska representationer, regler, beräkningar och output visar den dynamiska lösningen en kontinuerlig påverkan mellan två olika system (van Gelder, 1999).

Enligt Beer (2000) blir den dynamiska ansatsen som mest kraftfull tillsammans med en situerad ansats. Han menar att det finns många paralleller mellan synsätten och att det situerade synsättet redan finns inom AI:n idag. Beer (1995) pekar på stora möjligheter med den situerade ansatsen, men ifrågasätter dess möjligheter när det kommer till högre kognitiva förmågor, såsom språk och abstrakt resonerande.

## 2.4 Situerad AI

Traditionell AI har fokuserat på högre kognitiva förmågor såsom schackspelande och annan problemlösning. Brooks (1991a) menar att AI istället skall koncentrera sig på enklare förmågor och sedan bygga vidare på dessa, det vill säga arbeta ”bottom-up”. Han motiverar påståendet med att det har tagit mycket längre tid för evolutionen att få fram grundläggande beteenden såsom förflyttning och perception än vad det har tagit för de högre kognitiva förmågorna, såsom att resonera och att planera. Dessa grundläggande beteenden är dessutom nödvändiga för att de högre kognitiva förmågorna överhuvudtaget skall kunna användas. Om robotens förmåga att resonera tas bort, visar det sig snart att det är dynamiken i interaktionen mellan robot och miljö som bestämmer intelligensen.

Brooks (1991a) menar att robotarna skall befinna sig i den verkliga världen, det vill säga vara situerade. För att kunna reagera snabbt på ett givet stimuli fungerar det inte med en komplett inre modell av världen med stora beräkningskrav. Istället används världen som sin egen modell och agenten kan genom sin perception koncentrera sig på de relevanta delarna i omgivningen. Enligt Brooks (1991a) används inom traditionell AI symboler som kan ha betydelse för dem som skapat systemen men som inte kan grundas i den verkliga världen och därför knappast har någon betydelse för systemet (jämför Searles argument om det kinesiska rummet, avsnitt 2.1). Systemen deltar inte heller i världen på ett sätt som en riktig varelse utan hanterar enbart en inre modell av världen, med sina egna påhittade lagar. Det finns ingen distinktion mellan agentens kunskap och världen som den agerar inom och agenten har ofta en fullständig och perfekt bild av sin omgivning. Fokus har istället hamnat på vilken sorts representation som skall användas för att representera omvärlden (Brooks, 1991a).

Saknar agenten sensorer och motorer måste sensordata och handlingar passera en mellanhand, agenten interagerar då följaktligen inte med omgivningen direkt. Brooks (1991a) menar att detta kan vålla problem då mellanhanden tillrättalägger eller tolkar input- och outputdata på ett sätt som marginaliserar eller bortser från svåra – men viktiga – delar av agentens varande i världen. Ges systemen en kropp, det vill säga implementeras i en robot som agerar i en verklig värld, är det inte längre möjligt att bortse från eller marginalisera viktiga aspekter i interaktionen. Fattas delar kommer roboten inte att kunna utföra ett önskat beteende. Vidare anser Brooks (1991a) att en

kropp ger möjligheten till att fysiskt grunda interna representationer och ge mening till det som försiggår i systemet.

## 2.5 Inlärning för autonom robot

Nolfi och Floreano (2000) beskriver tre olika typer av inlärning: oövervakad inlärning (eng. unsupervised learning), övervakad inlärning (eng. supervised learning), samt inlärning genom belöning (eng. reinforcement learning). Med oövervakad inlärning anpassas nätverket enbart till den input som ges. Den här typen av inlärning används framför allt för att ta fram generella egenskaper hos en mängd element, dela in objekt i kategorier samt för att komprimera data (Nolfi & Floreano, 2000). Eftersom designern inte kan styra nätverkets utveckling mot ett mål (exempelvis att det skall lösa en viss uppgift), passar inte denna sorts inlärning för en autonom robot som skall lösa en specifik uppgift.

Vidare beskriver Nolfi och Floreano (2000) innebörden av övervakad inlärning. Denna sorts inlärning innebär att nätverket anpassas efter skillnaden mellan *önskad* output och *verklig* output, vilket kräver kännedom om vilken output som är korrekt givet en viss input. Den här typen av inlärning passar därför inte för autonoma robotar eftersom det oftast är okänt hur det utvecklade beteendet skall se ut (Nolfi och Floreano, 2000).

De mål som ges till autonoma robotar oftast är abstrakt beskrivna, exempelvis att den skall överleva så länge som möjligt eller att den skall korsa linjen för målzonen, snarare än specifikt beskrivna input-output-par (givet input {0.7, 1.2, 0.8, 1.1} ge output {0.3, 0.7})(Meeden, 1996). Meeden (1996) och Nolfi och Floreano (2000) menar att inlärning genom belöning är den sorts inlärning som passar bäst för agenter med abstrakt beskrivna mål.

Meeden (1996) gör en distinktion mellan lokal respektive global metod för inlärning genom belöning. Globala metoder utforskar hela sökrymden, vilket inte är fallet med lokala metoder. Meeden (1996) exemplifierar lokal metod med ”complementary reinforcement back-propagation”-inlärningsalgoritm (CRBP). Med denna algoritm förändras nätverket omedelbart under dess interaktion med omgivningen. Om nätverket givet en viss input ger en output som leder till en positiv respons belönas nätverket. Ger nätverket å andra sidan en output som leder till en negativ respons bestraffas det. Meeden (1996) förklarar att algoritmen leder till att handlingar som ger en positiv respons har stor sannolikhet att inträffa igen, medan handlingar som ger en negativ respons undviks.

Globala metoder exemplifieras av Meeden (1996) med en genetisk algoritm. Genetiska algoritmer är baserade på teorin om det naturliga urval som sker vid evolution. Givet en slumpmässigt genererad population av individer, där varje individ representerar en möjlig lösning på problemet, väljer algoritmen ut vissa av individerna. De individer som har en hög fitness (ett mått på hur väl individen löser uppgiften) har större sannolikhet att väljas ut. De utvalda individerna ligger sedan till grund för nästa generation som skapas genom mutation av de utvalda individerna (Meeden, 1996).

Meeden (1996) menar att genetiska algoritmer, tack vare möjligheten att utforska hela sökrymden, hittade mer tillförlitliga lösningar. Eftersom CRBP enbart söker i den närmaste omgivningen är riskerna stora att den fastnar i ett lokalt optima, det vill säga en lösning som inte är ett globalt optima. Hon pekar på metodernas möjlighet att komplettera varandra, där den globala metoden kan användas för att hitta

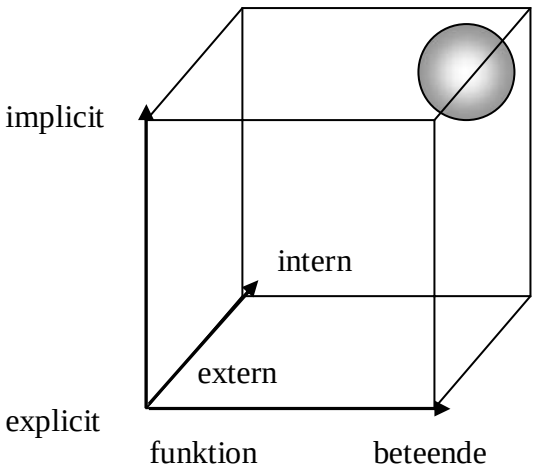
utgångslägen och den lokala metoden för att finjustera nätverket. Detta riskerar dock att bli extremt beräkningkrävande (Meeden, 1996).

## 2.6 Fitnessfunktion

Med hjälp av en fitnessfunktion räknas varje individs fitness ut.

Enligt Nolfi och Floreano (2000) beror agentens utveckling till stor del på hur fitnessfunktionen är utformad. Nolfi och Floreano (2000) föreslår fitnessrymden (eng. fitness space) som ett objektivt mått för att beskriva en fitnessfunktion. Fitnessrymden består av tre dimensioner, se Figur 6:

- *funktion* kontra *beteende*, om fitnessfunktionen värderar specifika funktionella tillstånd (specifika input-output-mappningar) eller om den värderar hela beteenden hos roboten. För att en agent skall utveckla ett gångbeteende kan en fitnessfunktion som värderar specifika funktionella tillstånd till exempel värdera frekvensen som noderna avfyrar i ett nätverk. En fitnessfunktion som värderar hela beteenden skulle istället exempelvis värdera hur långt agenten förflyttat sig.
- *explicit* kontra *implicit*, antalet variabler och begränsningar som ges av fitnessfunktionen. En explicit funktion använder många variabler och konstanter för att värdera agentens beteende, medan en implicit funktion enbart använder ett fåtal.
- *extern* kontra *intern*, om variablerna och begränsningarna i fitnessfunktionen beräknas med information tillgänglig för agenten eller inte. Det exakta avståndet mellan agenten och dess mål är en extern variabel eftersom den inte kan räknas ut enbart med den information agenten har. Aktivationen hos ljussensorerna är däremot en intern variabel eftersom den är tillgänglig för agenten.




**Figur 6: Fitnessrymdens tre dimensioner (efter Nolfi & Floreano, 2000, s.65).**

Nolfi och Floreano (2000) menar att om målet är att utveckla en robot som kan agera autonomt i en delvis okänd och oförutsägbar miljö, utan att en människa ingriper, bör fitnessfunktionen värdera hela beteenden, ha ett så litet antal variabler och begränsningar som möjligt samt ha variabler och begränsningar som är tillgängliga för roboten. Detta motsvarar det grå området i figurens övre högra hörn (Nolfi & Floreano, 2000).

## 2.7 Simulering

Jakobi (1997) menar att robotens utveckling går snabbare i en simulering än i verkligheten, men att en komplett simulering av en realistisk värld blir så pass beräkningsmässigt krävande och svår att designa att fördelen med simuleringen försvinner. Modelleras inte allt i miljön finns det dock risk för att viktiga aspekter av robotens interaktion med omgivningen försvinner, vilket leder till att kontrollarkitekturen inte på ett framgångsrikt sätt kan överföras från den robot som agerar i simuleringen till den robot som agerar i verkligheten. Jakobi (1997) beskriver en lösning för att förhindra alltför komplexa och beräkningsintensiva simuleringar och samtidigt behålla möjligheterna till att överföra den tränade kontrollarkitekturen från simulering till verklighet. Han menar att de delar av omgivningen som är nödvändiga för att roboten skall utveckla det önskade beteendet, den så kallade basmängden (eng. base set), skall modelleras på ett exakt sätt i simuleringen. Simulering kommer även att innehålla andra delar som inte är nödvändiga för att roboten skall utveckla beteendet, så kallade implementationsaspekter (eng. implementation aspects). Enligt Jakobi (1997) skall implementationsaspekterna slumpmässigt variera tillräckligt mycket för att roboten inte skall kunna lita på dem utan enbart kunna lita på basmängden, vilket kommer att leda till att robotens utvecklade beteende enbart beror på basmängden. Vidare menar Jakobi (1997) att det faktiskt inte går att modellera basmängden helt exakt, utan att även dessa istället måste variera slumpmässigt. Denna variation måste vara tillräckligt stor för att överbrygga skillnaderna mellan simulering och verklighet, men ändå så pass liten att roboten kan lita på basmängden (jämför med variationen för implementationsaspekterna).

Nolfi och Floreano (2000) kritiserar Jakobis (1997) ansats och förklarar att det är väldigt sällan som man faktiskt vet vilka aspekter av omgivningen som är viktig för robotens utveckling av ett visst beteende. De menar att:

> ”the robot-environment interactions that should be included in the base set of minimal simulations depend on the varieties of behavioral strategies that might be exploited during the evolutionary process which, in turn, cannot be known in advance by the designer” (Nolfi & Floreano, 2000, s.64).

Med Jakobis indelning kommer utvecklingen av robotens beteende att styras av designern i större grad än om utvecklingen får fritt spelrum i en simulering som är en komplett representation av verkligheten. Studiens mål är dock hur agenten hanterar en uppgift som kräver ett räkneliknande beteende, vilket antas kräva att beteendet kan generaliseras till obekanta situationer. Agentens beteende bör därför i så liten grad som möjligt vara beroende av aspekter som enbart finns i en specifik omgivning. För att det skall vara möjligt att generalisera beteendet till nya situationer antas det krävas vissa likheter mellan dessa och situationerna agenten redan mött. Det är dessa aspekter som är viktiga att modellera. Designerns styrning av beteendeutvecklingen ses därför inte som en nackdel för studien.

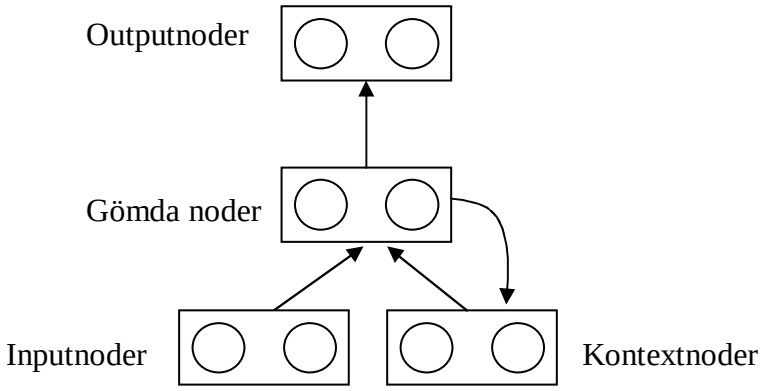
## 2.8 Tidigare arbeten

### 2.8.1 Kontrollarkitektur för räkning

Wiles och Elman (1995) tränar ett konnektionistiskt nätverk på en form av räkning. Nätverket tränas till att förutsäga ett enkelt kontextoberoende språk med sekvenser av formen $a^n b^n$. Uppgiften kräver att nätverket håller reda på hur många ”a” som funnits i sekvensen för att det skall matcha med lika många ”b”. Givet sekvensen ”aaab” skall

alltså systemet förutsäga resten av sekvensen ”bba”. Nätverket tränas på sekvenser upp till n=11, men lyckas generalisera upp till n=21.

För att lösa uppgiften använder Wiles och Elman (1995) ett enkelt återkopplat nätverk (eng. Simple Recurrent Network), se Figur 7. Elman (1990) förklarar att denna typ av nätverksarkitektur har kopplingar tillbaka till ett tidigare lager. Dessa kopplingar ger nätverket möjligheten att se sin tidigare aktivation och anpassa efterföljande aktivation till detta. Med andra ord: nätverket får ett minne av tidigare tillstånd, ett korttidsminne[2].




**Figur 7: Enkelt återkopplat nätverk (efter Ziemke, 2000, s.44).**

Återkopplingen leder till att aktivationen i de gömda noderna vid tidssteg t+1 påverkas av aktivationen vid tidssteg t, därför behöver inte den input som vid tidssteg t gav en viss output ge samma output vid tidssteg t+1 (Ziemke, 2000). Detta är att jämföra med ett reaktivt nätverk som saknar återkoppling, då samma input alltid ger samma output.

Enligt Wiles och Elmans (1995) är en intuitiv lösning på problemet att en av noderna används som räknare och får en högre aktivation för varje ”a” som ges som input. Ett stort antal ”a” skulle alltså motsvaras av en hög aktivation i en viss nod. Under analysen visar det sig dock att en av de gömda nodernas aktivation kretsar runt (högre och lägre) ett visst värde och för varje a som presenteras närmar sig aktivationen värdet. Wiles och Elman (1995) liknar det vid att dra upp en fjäder. När det första ”b” presenteras för nätverket tar den andra gömda noden över och ”rullar ut fjädern” under en tid som motsvarar antalet ”a”. Wiles och Elman (1995) understryker att resultatet visar att det som intuitivt kräver en räknare kan lösas av mekanismer som delar vissa, men inte alla, egenskaper som finns hos en räknare.

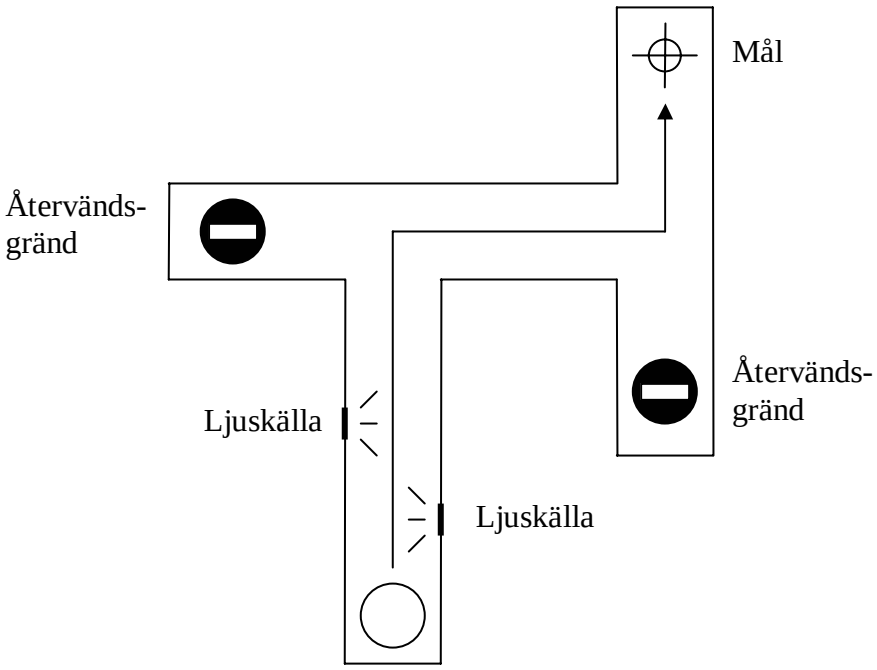
Dorffner (1997) menar dock att angreppssättet med förberedda representationer och utvalda träningsexempel som input, lämnar lite kvar till nätverket att lösa (se avsnitt 2.2). Kritik grundad på samma orsaker finns också i Brooks (1991a) beskrivning av den situerade agenten. Han menar att osituerade lösningar kan vålla problem då de marginaliserar viktiga delar av agentens interaktion med omgivningen (se avsnitt 2.4). Wiles och Elman (1995) väljer att enbart inrikta sig mot en funktion, räkning, och helt förbise svårigheterna med funktionerna för exempelvis perception och handlande. Inputen i deras experiment består av två distinka kategorier, a respektive b, som

[2] Se Baddeley (1999) för definition av korttidsminne.

presenteras för nätverket under ett bestämt antal tidssteg, utan något av det brus som finns i vår omgivning. Nätverkets output tolkas av en mellanhand för att avgöra om det lyckats lösa uppgiften. Systemets förutsättningar är alltså orealistiska och kan knappas generaliseras till en agent som agerar i en verklig miljö.

### 2.8.2 Kontrollarkitektur för fördröjt svar

Thieme (2002) beskriver ett situerat experiment där en simulerad robot skall färdas en bestämd väg genom en labyrint med två T-formade korsningar, se Figur 8. Robotens uppgift är att nå ett mål inom ett begränsat antal tidssteg, utan att hamna i en återvändsgränd. Thieme (2002) förklarar att det finns två lampor i början av labyrinten och att dessa, beroende på om de är placerade på vänster eller höger sida av korridoren, visar vilken väg roboten skall välja. Första ljuskällan visar hur roboten skall svänga i den första korsningen och den andra ljuskällan visar hur roboten skall svänga i den andra korsningen. Lamporna är placerade på tillräckligt stort avstånd för att roboten inte skall uppfatta lampornas ljusstimuli från någon av korsningarna. Enligt Ziemke och Thieme (under tryckning) är denna form av uppgift, där det krävs att agenten väntar med att svara på givet stimuli, vanligt förekommande för att undersöka korttidsminnet. Det antas enligt Ziemke och Thieme (under tryckning) att agenten kommer ihåg nödvändig information om stimulit under en viss period, det vill säga att den har korttidsminne.

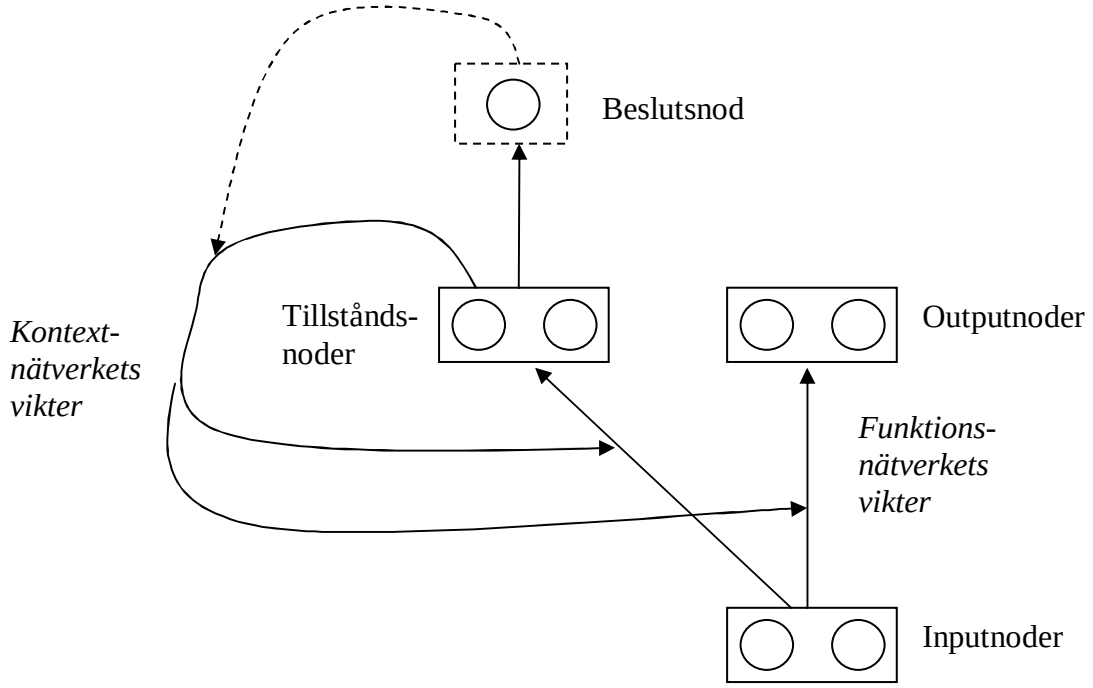



**Figur 8: Labyrint med två T-formade korsningar (efter Thieme, 2002, s.10).**

Thieme (2002) jämför flera olika kontrollarkitekturer, däribland ett enkelt återkopplat nätverk (se avsnitt 2.8.1) och ett ”Extended Sequential Cascaded Network” (ESCN) för att se hur väl de löser uppgiften.

Enligt Ziemke (2000) kan ett ESCN beskrivas som två nätverk: ett funktionsnätverk som mappar input till output (sensomotorisk mappning) och ett kontextnätverk som ”tar” tillståndsnodernas aktivation och beräknar kopplingarnas vikter för nästa tidssteg, se Figur 9. Vikterna i funktionsnätverket förändras enbart när beslutsnoden är aktiverad. Beslutsnoden avgör alltså när tillståndsnodernas aktivation skall påverka

kopplingarnas vikter i funktionsnätverket. Aktiveras inte beslutsnoden förblir vikterna i nätverket oförändrade (Ziemke, 2000).




**Figur 9: ”Extended Sequential Cascaded Network” (efter Ziemke, 2000, s.109).**

Funktionsnätverkets sensomotoriska mappning beror således på nätverkets interna tillstånd (aktivationen i tillståndsnoderna) vid tidigare tidssteg. Den input som vid ett tillfälle givit en viss output, kan alltså vid ett senare tillfälle vara källa till någon helt annan output. Enligt Ziemke (2000) gör en beslutsnod att nätverket själv aktivt kan bestämma när den sensomotoriska mappningen i funktionsnätverket skall förändras. Vikterna i funktionsnätverket behöver alltså inte förändras i varje tidssteg. Ziemke (1999) förklarar att denna typ av arkitektur möjliggör en virtuell modularitet, eftersom den genom självorganisering anpassar sitt beteende genom att förändra sina sensomotoriska mappningar på ett sätt som om den hade olika moduler för olika situationer.

I Thiemes (2002) experiment ges inga förutbestämda kategorier som input till arkitekturen. Agenten agerar i en simulerad värld och dess beteende och interna tillstånd självorganiseras under kontrollarkitekturens inlärning av uppgiften. Eftersom agenten är situerad görs inga överdrivna antaganden om andra moduler eller om perfekt input och output (se avsnitt 2.4).

Thieme (2002) förklarar att om en agent använder representationer i Palmers (1978) mening kommer representationerna att fungera som en ersättare för det stimuli agenten får från ljuskällorna. Representationerna kommer att finnas lagrade inom agenten, i någon sorts minne och användas först när agenten når korsningen och då visa vilken väg agenten skall välja. Thieme (2002) menar att det enligt detta synsätt måste finnas en 1:1-korrespondens mellan en representation och respektive stimuli (se avsnitt 2.1), det vill säga att en representation av en ljuskälla *enbart* är en representation av en ljuskälla och ingenting mer.

Resultatet visar att ett ESCN presterar bäst av arkitekturerna och löser uppgiften i 99,4 % av fallen. Detta kan jämföras med ett enkelt återkopplat nätverk som löser

uppgiften i 92,5 % av fallen.  Ziemke och Thieme (under tryckning) belyser resultaten av ESCN i Thieme (2002). De beskriver att agenten med ett ESCN som kontrollarkitektur, så långt som möjligt är helt reaktiv och endast byter den sensomotoriska mappningen vid något enstaka tillfälle. De sensomotoriska mappningarna korresponderar mot ett modulärt beteende som kontrollerar agenten, men Ziemke och Thieme (under tryckning) menar att de knappast motsvarar en traditionell representation. De påvisar att bytet av de sensomotoriska mappningarna oftast inte utfördes i samband med ljusstimulit och menar att det därför inte kan ses som en representation av ljuskällan i Palmers mening, det vill säga: det råder inte en 1:1-korrespondens mellan intern representation och externt objekt. Ziemke och Thieme (under tryckning) påstår ändå att agenten använder sig av ett korttidsminne, men att minnet inte är en behållare för traditionella representationer, utan snarare fungerar som en ledsagare för kommande beteende. Detta stämmer väl överens med Dorffners (1997) interaktionsbaserade representationer (se avsnitt 2.2).

Att nätverket klarade av att lösa uppgiften med två korsningar, visar på begränsade tendenser till att kunna räkna. Det krävs av agenten att den minns vilken sida ljuskällorna finns samt ordningen mellan dessa. Fokus för Thieme (2002) är dock inte på möjligheterna för ett ESCN att räkna och resultatet har därför inte analyserats utifrån denna utgångspunkt.

# 3 Problemprecisering

Målet med studien är att undersöka hur en agent med kontrollarkitekturen ”Extended Sequential Cascaded Network” (ESCN) hanterar en uppgift som kräver ett räkneliknande beteende. Ett räkneliknande beteende definieras för studien att vara ett beteende som är beroende av antalet specifika stimuli som presenterats skilt från den situation som fordrar responsen.

Målet delas upp i två delmål. Det första delmålet är att mäta pålitligheten i agentens beteende. Denna mäts i situationer med ett lika stort antal stimuli som under träningen.

Det andra delmålet är att analysera hur de interna tillstånden vägleder agenten med fokus på om den använder traditionell representation för att lösa uppgiften. Detta kommer att kontrolleras med avseende på kopplingarnas vikter, aktivationen i noderna samt agentens beteende ur ett distalt perspektiv. Utgångspunkterna kommer att vara traditionella representationer (Palmer, 1978), interaktionsbaserade representationer (Dorffners, 1997) samt dynamiska system (Beer, 1995, 2000; Gelder, 1999).

Thiemes (2002) fokus ligger utanför den situerade agentens möjlighet till räkning och Wiles och Elman (1995) fokuserar på en osituerad agent utan förankring i en realistisk värld. En naturlig fortsättning på dessa arbeten är därför att undersöka hur den kontrollarkitektur som används i Thieme (2002) hanterar en uppgift som kräver ett räkneliknande beteende. Ett sådant experiment skiljer sig från Wiles och Elman (1995) i och med utgångspunkten från en situerad agent och från Thieme (2002) i och med fokus på ett räkneliknande beteende.

## 3.1 Avgränsning

I arbetet kommer enbart en enkel form av räkning, motsvarande den i Wiles och Elman (1995), att tas upp. Agenten ställs alltså inför en uppgift som vid ett eller flera tillfällen antas kräva addition eller subtraktion med 1.

Uppgiften kommer att hållas inom enkla simulerade världar. För att ha möjligheten att överföra kontrollarkitekturen från den simulerade till den verkliga världen kommer implementationen av simuleringen att följa Jakobis (1997) idéer om uppdelningen av den simulerade världen i basmängd respektive implementationsaspekter. Nolfi och Floreano (2000) visar dock på riskerna med ansatsen och menar att den kan leda till att designerns egna förförståelse leder utvecklingen av beteendet. Med möjligheten till en snabb implementering av simulatorn samt underlättandet vid insamling och analys av data anses dock fördelarna med Jakobis ansats överväga nackdelarna.

# 4 Experimentet

Liksom i Thiemes (2002) arbete genomförs studien med hjälp av experiment som implementeras i en kheperasimulator (robotsimulator).

I avsnitt 4.1 beskrivs experimentets upplägg. Här tas det upp vilken kheperasimulator som kommer att användas, hur uppgiften ser ut, en beskrivning av kontrollarkitekturen samt hur agenten kommer att tränas.

Resultatet består av en procentsats för antalet lyckade försök att lösa uppgiften, kopplingarnas vikter i nätverket, nodernas aktivation under försöken att lösa uppgifterna samt bilder av hur agenten har rört sig i miljön. I avsnitt 4.2 beskrivs hur resultatet kommer att analyseras.

Under utvecklingen av agentens beteende uppstod komplikationer. Agenten visade inga tendenser till att lära sig ett räkneliknande beteende och presterade dåligt på testerna. För att få en indikation på vad som kunde vara fel, om det bakomliggande problemet grundades i kontrollarkitekturen, gjordes ett kompletterande experiment med en annan nätverksarkitektur. Det kompletterande experimentet beskrivs i avsnitt 4.3.

## 4.1 Experimentets upplägg

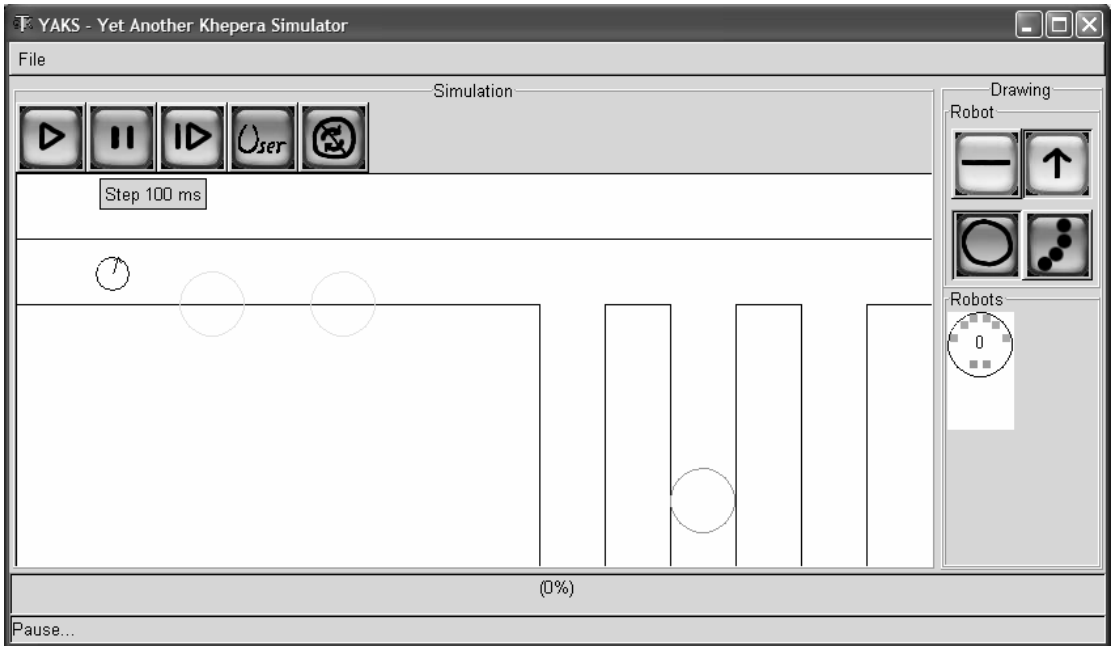
Den simulerade roboten tränas först på att lösa en uppgift och sedan testas den på hur väl den löser uppgiften.

### 4.1.1 Agentens uppgift

Agenten placeras i en simulerad värld bestående av en korridor med flera avtagande gångar, se Figur 1. I början av korridoren finns ett varierande antal lampor från vilka agenten får ljusstimuli. Målet för agenten är att förflytta sig över en linje som aktiverar marksensorn. Linjen finns endast i den gång som motsvaras av antalet lampor och agenten får inte gå in i en gång som saknar linjen. En lampa motsvarar den första gången, två lampor motsvarar den andra gången och så vidare. Lampornas ljusstimuli når inte agenten när den befinner sig vid gångarna, agenten behöver alltså en mekanism som vägleder dess beteende vid denna del av uppgiften.

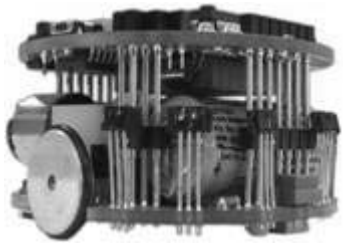
### 4.1.2 Yet Another Khepera Simulator (YAKS)

Som simulator för experimentet används ”Yet Another Khepera Simulator” (YAKS), se Figur 10.




**Figur 10: Gränssnitt för Yet Another Khepera Simulator.**

YAKS kan i teorin simulera ett obegränsat antal kheperarobotar som interagerar med en obegränsat stor värld (Carlsson & Ziemke, 2001). Mondada, Franzi och Ienne (1993) beskriver att kheperaroboten har en cylindrisk form med diametern 55 millimeter och höjden 30 millimeter, se Figur 11. Roboten förflyttar sig med hjälp av två hjul, vilka kan rotera både framåt och bakåt oberoende av varandra. I grundutförandet får roboten input från 8 infraröda sensorer, 6 placerade på framsidan och 2 på baksidan. Dessa kan både mäta ljusstyrkan de utsätts för samt uppskatta avståndet till närmaste hinder (Mondada m.fl.,1993).



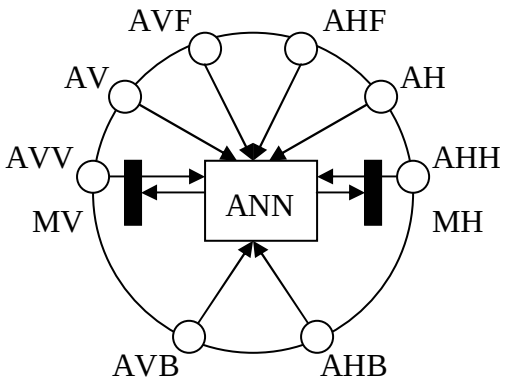
**Figur 11: Kheperarobot i grundutförande.**

Enligt Mondada m.fl. (1993) kan robotens kontrollsystem exekveras i en dator som läser in sensorvärden och styr motorerna, samtidigt som roboten interagerar med omgivningen. Eftersom all data från såväl sensorer som motorer då finns tillgänglig på datorn blir det enkelt att analysera robotens beteende. Vidare pekar Mondada m.fl. (1993) på möjligheten till att utveckla kontrollalgoritmer i en dator och sedan ladda ned dem i kheperaroboten. Roboten kan då köras autonomt, oberoende av utvecklingsmiljön. Mondada m.fl. (1993) lyfter också fram att robotens ringa storlek medför möjligheten till att utföra snabba experiment som endast kräver en liten yta. Vidare menar de att robotens modulära uppbyggd tillåter att extra moduler för sensorer och motorer läggs till. Dessa fördelar har tidigare robotar saknat (Mondada m.fl., 1993).

Kheperasimulatorn YAKS har flera egenskaper som gör att den passar för studien: den är gratis, den har ett tydligt grafiskt gränssnitt, vikterna för den tränade kontrollarkitekturen kan överföras mellan flera olika plattformar (bland annat mellan Windows XP och UNIX, vilka båda används i experimentet) och den är ”snabb” (Carlsson & Ziemke, 2001). Vidare är simulatorn förberedd för ESCN (N. Bergfeldt, personlig kontakt, 14 mars, 2003), vilket minskar studiens implementationsdel.

### 4.1.3 Agentens kontrollarkitekturer

I experimentet kommer ett ESCN att användas som kontrollarkitektur (se avsnitt 2.8). Som input till kontrollarkitekturen finns tre olika typer av sensorer: *avståndssensor* som reagerar på objekt, *ljussensor* som reagerar på ljus samt *marksensor* som aktiveras när roboten passerar den linje som markerar en målzon. Kontrollarkitekturen får sin input från 8 avståndssensorer, 8 ljussensorer och 1 marksensor, se Figur 12. Vidare har kontrollarkitekturen 2 outputnoder som kontrollerar vänster respektive höger motor. Nodernas värden motsvarar hastigheten i motorerna, från full fart bakåt till full fart framåt. Totalt kommer kontrollarkitekturen att ha 17 inputnoder, 3 kontextnoder, 1 beslutsnod samt 2 outputnoder.




**Figur 12: Figur ovanifrån över inputsensorer och outputmotorer för den simulerade kheperaroboten (efter Thieme, 2002, s.11). Beteckning A refererar till avstånds- och ljussensor: vänster bak (AVB), vänster-vänster (AVV), vänster (AV), vänster fram (AVF), höger fram (AHF), höger (AH), höger-höger (AHH) och höger bak (AHB). Beteckning M refererar till motor: vänster (MV) och höger (MH). Marksensorn (visas inte i figur) finns vid robotens mittpunkt.**

Enligt Ziemke (2000) finns inte marksensorn implementerad i en riktig kheperarobot, men han pekar på möjligheter till att bygga ut roboten med denna funktion. Denna begränsning kommer dock inte utgöra något problem för studien eftersom det i studien används en simulerad robot vars kontrollarkitektur inte skall implementeras i en verklig kheperarobot.

### 4.1.4 Implementation

I experimentet kommer agentens inlärning att ske med hjälp av genetiska algoritmer, vilket är en global metod för inlärning genom belöning (se avsnitt 2.5). Under en epok får agenten ett försök till att lösa problemet. En epok avslutas efter 1000 tidssteg, när agenten korsar linjen för målzonen, när agenten går in i en gång där inte målzonen finns eller när agenten förflyttar sig för långt bort i korridoren (motsvaras av tecknen för enkelriktat i Figur 1).

Varje individ tränas på upp till 3 lampor maximalt 4 gånger. Första epoken finns 1 lampa, andra epoken finns 2 lampor, tredje epoken 3 lampor, fjärde epoken 1 lampa och så vidare. Träningen för varje individ avbryts dock vid det första misslyckandet, det vill säga: om individen klarar av den första epoken men inte den andra, får den inte delta i en tredje (eller senare) epok. En individ deltar alltså i mellan 1 – 12 epoker.

Varje population består av 100 individer. Vikterna samt aktivationen i kontextnoderna hos individerna i den första generationen slumpas ut med en gaussisk distribution med standardavvikelsen 2. För att skapa populationer med en stor spridning kommer 9 av de 10 individer som väljs ut till nästa generation utses genom tävling, där två individer slumpmässigt väljs ut ur populationen och den av de två som har högst fitnesspoäng går vidare till nästa generation. För att hålla kvar individen med det för tillfället mest gynnsamma beteendet väljs den 10:e individen ut på basis av högsta fitnesspoäng. Varje vald individ utgör sedan grunden för 10 nya individer i nästa generation. De utvalda individernas vikter samt aktivation i kontextnoderna muteras med en gaussisk distribution med standardavvikelsen 2.

Totalt kommer 4900 generationer att utvecklas. Den bästa individen i den sista generationen kommer att testas 1000 gånger per variant av uppgiften. Sammanlagt kommer agenten alltså att få 3000 försök.

Agenten i studien har enbart kunskap om de delar av domänen som den uppfattar genom sina sensorer, vilket innebär att den agerar i en (för agenten) delvis okänd och oförutsägbar omgivning. Vidare bör utvecklingen påverkas av designerns i så liten grad som möjligt. Enligt Nolfi och Floreano (2000) bör utvecklingen av en agent med dessa förutsättningar ske med en fitnessfunktion som värderar hela beteenden, har ett så litet antal variabler och begränsningar som möjligt samt ha variabler och begränsningar som är tillgängliga för roboten (se avsnitt 2.6). I studien används därför följande fitnessfunktion:

$$\sum_{i=1}^{n} f_i(x) \begin{cases} 1 \text{ om agenten korsar linjen för målzonen} \\ 0 \text{ annars} \end{cases} \qquad \textit{(ekvation 1)}$$

Fitnessfunktionen summerar resultatet för de epoker ($n$) individen ($x$) deltar i, vilket ger mellan 0 – 12 fitnesspoäng till varje individ. Den värderar hela beteendet genom att enbart ge poäng när agenten har nått målet. Vidare används enbart en variabel, vilken också är tillgänglig för agenten själv.

En precisering av vilka element som modelleras i simuleringen antas vara nödvändig med hänsyn till omfattningen hos en fullständig implementation av en verklig situation (se avsnitt 2.7). Denna precisering är avgörande för vilket beteende agenten utvecklar och kan innebära att den blir beroende av aspekter som är unika för implementationen och följaktligen inte är lika betydande i en verklig situation. För att undvika ett sådant beteende väljs i enlighet med Jakobi (1997) att enbart ge agenten möjlighet till att lita på de element som antas återkomma i varje situation. Detta har givit följande kategorisering mellan de delar som agenten kan lita på (basmängden) respektive de delar som agenten inte kan lita på (implementationsaspekter):

I) *Basmängden* för simuleringen är antalet lampor i korridoren samt antalet gångar som leder ut från korridoren. Den sistnämnda måste vara lika med eller högre än antalet lampor för att roboten skall kunna utveckla beteendet. Robotens sensorvärden kommer att påverkas av brus, +/- 5% av det totala sensorvärdet.

II) *Implementationsaspekter* för simuleringen är robotens startposition och startvinkel, korridorens bredd, avståndet mellan korridorens början och den första lampan, avståndet mellan varje lampa, gångarnas bredd, avståndet mellan gångarna samt avståndet mellan den sista lampan och den första gången. De flesta implementationsaspekter kommer att slumpas ut inom intervallet $Y < X < 2Y$, där Y är det lägsta värdet som kan antas. För korridorens bredd, gångarnas bredd samt avståndet mellan gångarna gäller att Y är lika med robotens diameter. För avståndet mellan varje lampa, avståndet mellan korridorens början och den första lampan samt avståndet mellan den sista lampan och den första gången gäller att Y är lika med två lampors radie tillsammans. De implementationsaspekter som skiljer sig från detta är robotens startposition samt startvinkel. Startpositionen slumpas ut till en plats till vänster om det första ljuset och startvinkeln slumpas ut till ett tal mellan 0-360 (grader). Det antas att variationerna för implementationsaspekterna är tillräckligt stora för att agenten inte skall kunna utveckla ett beteende som är beroende av dessa.

Alla individer kommer att agera i miljöer med samma slumpmässiga variation, det vill säga både basmängd och implementationsaspekter kommer att vara identiska för varje individ givet en viss epok.

### 4.1.5 Validering av implementation

För att kontrollera ESCN-implementationen i YAKS replikerades det enklaste experimentet i Thieme (2002). Agenten placerades i en korridor med en lampa i början av korridoren och en korsning i slutet av korridoren. Beroende på vilken sida av korridoren lampan fanns skulle agenten svänga åt vänster eller höger i korsningen. Givet en lampa på korridorens vänstra sida skulle agenten svänga åt vänster i korsningen. Givet en lampa på den högra sidan skulle agenten svänga åt höger i korsningen.

Agenten utvecklades med en genetisk algoritm, där fitnessfunktionen gav en poäng när agenten gjorde rätt vägval. Varje generation bestod av 100 individer. Varje individ tränades på upp till 6 epoker i serien: lampa på vänster sida, lampa på höger sida, lampa på vänster sida, lampa på höger sida, lampa på vänster sida och lampa på höger sida. Vid det första misslyckade försöket avbröts träningen för individen. En individ kunde alltså få mellan 0 – 6 fitnesspoäng.

Totalt tränades agenten i 160 generationer. Sedan testades agenten 1000 gånger, 500 gånger med en lampa på vänster sida respektive 500 gånger med en lampa på höger sida. Agenten löste uppgiften 987 gånger av dessa, det vill säga i 98,7 % av fallen.

Detta visar att en agent med ESCN-implementationen i YAKS kan utvecklas till att lösa enklare uppgifter som kräver ett fördröjt svar. För studien antas därför att ESCN-implementationen i YAKS är tillräckligt felfri för att användas som simuleringsverktyg.

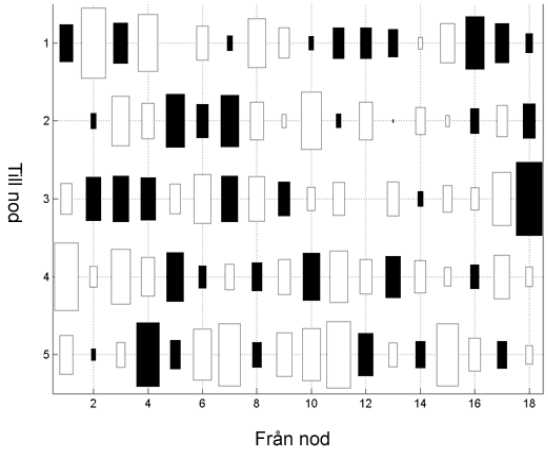
## 4.2 Utvärdering av experimentet

Studiens mål har delats upp i två delmål. Det första delmålet är att mäta hur pålitlig agentens lösning av uppgiften är. Detta kommer att göras med hjälp av en procentsats:

$$F = \frac{M}{T} \qquad \textit{(ekvation 2)}$$

M är antalet framgångsrikt lösta uppgifter, T är antalet gjorda försök och F är andelen framgångsrikt lösta uppgifter.

Det andra delmålet är hur de interna tillstånden vägleder agenten med fokus på om den använder traditionell representation för att lösa uppgiften. Detta kontrolleras genom att analysera de interna tillstånden samt agentens interaktion med omgivningen ur ett distalt perspektiv. Kopplingarnas vikter beskrivs med Hinton-diagram, se Figur 13. Ett Hinton-diagram är en grafisk beskrivning av vikternas magnitud i nätverket. X-axeln i diagrammet refererar till noden från vilken kopplingen går och Y-axeln refererar till noden till vilken kopplingen går. I diagrammet finns en ruta för varje koppling, ju större ruta desto extremare värde. En vit ruta betyder positiv koppling och en svart ruta betyder negativ koppling. Exempelvis innebär alltså en stor svart ruta att kopplingen har ett starkt negativt värde.




**Figur 13: Exempel på ett Hinton-diagram.**

Vidare kommer aktivation i noderna att beskrivas med aktivationskurvor, som vid varje tidssteg visar aktivationen för varje nod i nätverket (input-, output-, kontext- och beslutsnoder). Agentens beteende kommer att övervakas genom en översiktsbild av omgivningen där agentens väg finns markerad.

## 4.3 Kompletterande experiment

Resultatet från experimentet visade på en agent som inte utvecklat ett räkneliknande beteende och enbart klarade av att lösa uppgiften med 1 lampa. För att få en indikation på vad som kunde vara fel, om det bakomliggande problemet grundades i kontrollarkitekturen, gjordes ett kompletterande experiment med en annan nätverksarkitektur.

I detta kompletterande experiment tränades en agent med ett enkelt återkopplat nätverk som kontrollarkitektur (se avsnitt 2.8.1). I denna typ av arkitektur hålls aktivationen från tidigare tillstånd kvar och påverkar nätverkets kommande tillstånd.

Uppgiften var nästan identisk med den för ESCN, med skillnaden att färre delar av miljön varierades. Eftersom en agent i en statisk miljö har möjligheten att utveckla ett beteende som är beroende av andra aspekter än vad Jakobi (1997) beskriver som basmängd, kan den utveckla ett beteende som är beroende av delar som är specifika för miljön den utvecklas i. Ett sådant beteende skulle inte framgångsrikt kunna överföras från en simulerad till en verklig miljö. Möjligheten att utveckla ett beteende som kan vara beroende av fler aspekter än de som designern bestämt antas dock öka

agentens möjligheter till att framgångsrikt lösa uppgiften. Klarar inte heller ett enkelt återkopplat nätverk av att lösa uppgiften trots färre variationer i miljön tyder det på att problemet inte uteslutande är knutet till ESCN-arkitekturen.

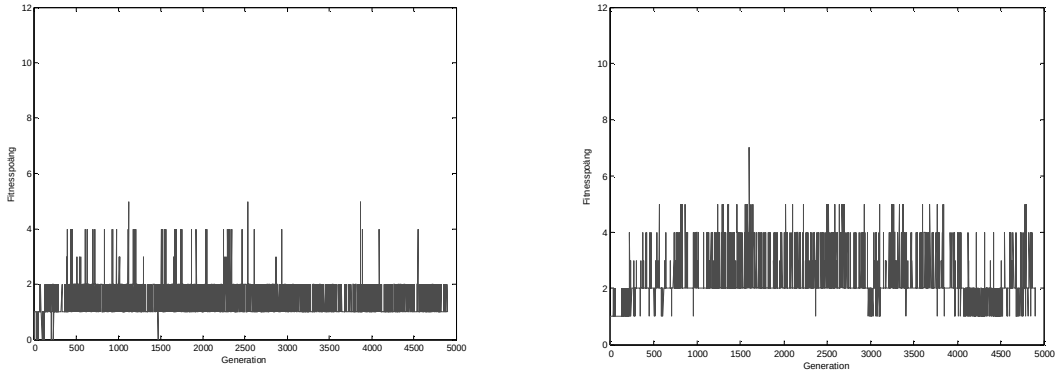
Variationerna som finns i det kompletterande experimentet begränsas till att enbart gälla agentens startposition och startvinkel. Startpositionen slumpas ut till en plats till vänster om det första ljuset och startvinkeln slumpas ut till ett tal mellan 0-360 (grader). Utvecklingen av agenten sker i övrigt på samma sätt som för ESCN.

# 5 Resultat och analys

I Avsnitt 5.1 beskrivs agentens utveckling. I avsnitt 5.2 beskrivs pålitligheten i agentens hantering av uppgiften. I avsnitt 5.3 beskrivs hur de interna tillstånden vägleder agenten i dess hantering av uppgiften.

## 5.1 Agentens utveckling

Individen med högst fitnesspoäng i varje generation och nätverksarkitektur beskrivs med kurvorna nedan (Figur 14).



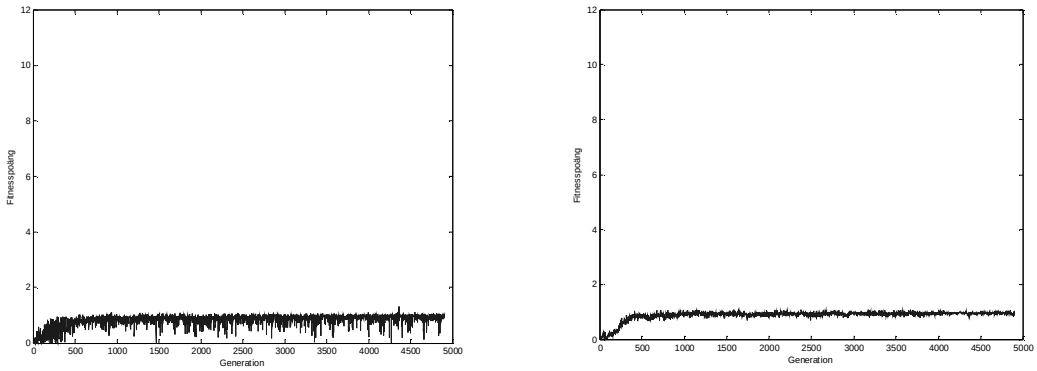

(a) (b)

**Figur 14: Kurvor över högsta fitnesspoäng i varje generation. (a) beskriver utvecklingen för ESCN och (b) beskriver utvecklingen för enkelt återkopplat nätverk.**

Fitnesskurvorna visar på en instabil utveckling hos agenten för båda nätverksarkitekturerna. Det enkla återkopplade nätverket har högre toppar och mindre djupa dalar än ESCN. De 4000 första generationerna håller sig agenten med ett enkelt återkopplat nätverk över 2 fitnesspoäng, det vill säga agenten klarar av uppgiften upp till 2 lampor. Med ett ESCN varierar genomgående agentens resultat från att den klarar av att lösa situationer med 1 lampa till att den klarar av att lösa situationer med 2 lampor.

Ingen av individerna hamnar i närheten av den maximala fitnesspoängen på 12 poäng. De toppar som finns är isolerade och visar på en lösning som inte håller över flera generationer.

Genomsnittlig fitnesspoäng för de 10 utvalda individerna i varje generation och nätverksarkitektur beskrivs med kurvorna nedan (Figur 15).




(a) (b)

**Figur 15: Kurvor över genomsnittlig fitnesspoäng hos de 10 utvalda individerna i varje generation. (a) beskriver utvecklingen för ESCN och (b) beskriver utvecklingen för enkelt återkopplat nätverk.**

## 5.2 Pålitlighet

Pålitligheten i agentens lösning av uppgiften givet de olika nätverksarkitekturerna sammanfattas i Tabell 1. 1000 försök gavs per nätverksarkitektur och situation (antal lampor).

| *Antal lampor* | *ESCN* | *Enkelt återkopplat nätverk* |
|---|---|---|
| 1 | 83,1 % | 96,1 % |
| 2 | 0,6 % | 1,7 % |
| 3 | 0,1 % | 0 % |

**Tabell 1: Andelen lyckade försök per situation.**

I försöken med 1 lampa når agenten med ett ESCN som kontrollarkitektur målzonen i 83,1 % av försöken. Detta är ett signifikant lägre resultat än de 96,1 % agenten med ett enkelt återkopplat nätverk som kontrollarkitektur presterade.

I de fall agenten misslyckas beror det på att den ”fastnar” vid den första gångens övre bortre hörn. Detta kan härledas från att agenten närmar sig den första gången i olika stora vinklar beroende på startposition och startvinkel (som slumpats ut). Är startposition och startvinkel ogynnsamma misslyckas agenten med att nå målzonen i den första gången.

Agentens beteende håller i sig oberoende av antalet lampor. Detta leder till att den enbart i enstaka fall lyckas lösa uppgifter med 2 eller 3 lampor. De fall där den lyckas lösa uppgifter med fler än 1 lampa kan alltså utifrån agenten ses som misslyckade försök till att gå in i den första gången, där agentens startposition och startvinkel har varit så pass ”ogynnsamma” att den helt missat den första gången och istället gått in i den andra eller tredje gången.

Sannolikheten för att agenten gör ett sådant ”misslyckat försök” och går in i en gång som av en tillfällighet råkar vara den rätta är liten men uppenbarligen inte obefintlig.

Dock sjunker andelen lyckade försök med antalet lampor. Detta kan tolkas som att ju närmare gången befinner sig den första gången, desto större är sannolikheten att agenten går in i den vid ett ”misslyckat försök”.

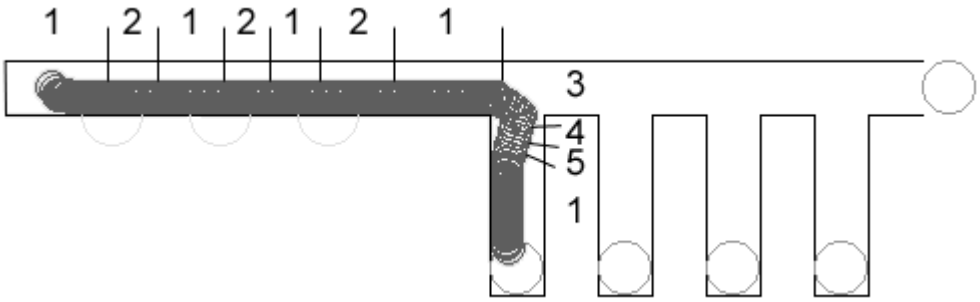
Att det inte utvecklats en individ som klarar av att lösa 100 % av försöken med 1 lampa kan tillskrivas samma slump som tidigare nämnts. Individer som gör ”misslyckade försök” och går in i gångar som av en tillfällighet råkar vara rätt får en högre poäng än individer som enbart följer den högra väggen. Därför konkurreras individerna med ett ”följa den högra väggen-beteende” ut av individer med en sensomotorisk mappning som råkar vara bättre anpassad för de slumpmässiga variablerna startposition och startvinkel. Detta stöds av att kurvorna för maximal och genomsnittlig fitness som visar att agenten (oberoende av kontrollarkitektur) inte utvecklar ett beteende som håller sig över flera generationer.

## 5.3 Agentens hantering av uppgiften

Först beskrivs hur agenten med ett ESCN som kontrollarkitektur hanterar uppgiften, sedan ges en beskrivning av hur en agent med ett enkelt återkopplat nätverk hanterar uppgiften.

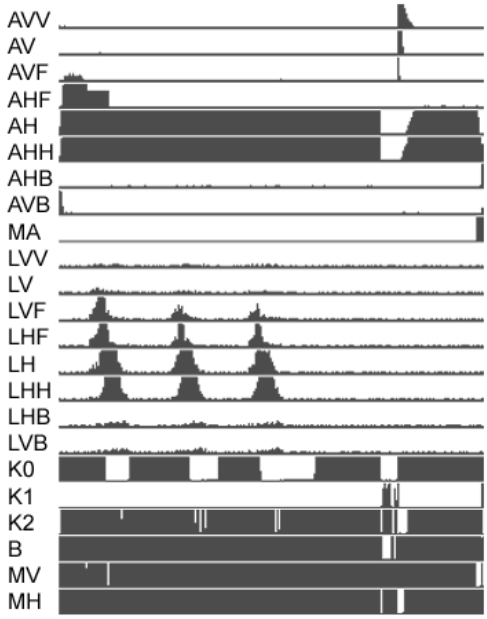
### 5.3.1 ESCN

Eftersom agenten med ett ESCN som kontrollarkitektur har samma beteende oberoende av antalet lampor kommer enbart *en* situation att analyseras för att beskriva beteendet. Situationen som valts innehåller 3 lampor, vilket alltså är den mest komplexa situation agenten tränats på, se Figur 16.




**Figur 16: Översiktsbild av agentens beteende i en situation med 3 lampor. Siffrorna refererar till den sensomotoriska mappning som används under sträckan.**

Kontrollarkitekturens noder beskrivs med aktivationskurvor där en kurva motsvarar aktivationen i en nod under agentens interaktion med omgivningen, se Figur 17. Beteckningen A refererar till avståndssensor: vänster-vänster (AVV), vänster (AV), vänster fram (AVF), höger fram (AHF), höger (AH), höger-höger (AHH), höger bak (AHB) och vänster bak (AVB). Beteckningen MA refererar till marksensor (MA). Beteckningen L refererar till ljussensor: vänster-vänster (LVV), vänster (LV), vänster fram (LVF), höger fram (LHF), höger (LH), höger-höger (LHH), höger bak (LHB) och vänster bak (LVB). Beteckningen K refererar till kontextnod, totalt 3 stycken (K0, K1, K2). Beteckningen B refererar till beslutsnod (B). Beteckningen M refererar till motor: vänster (MV) och höger (MH).

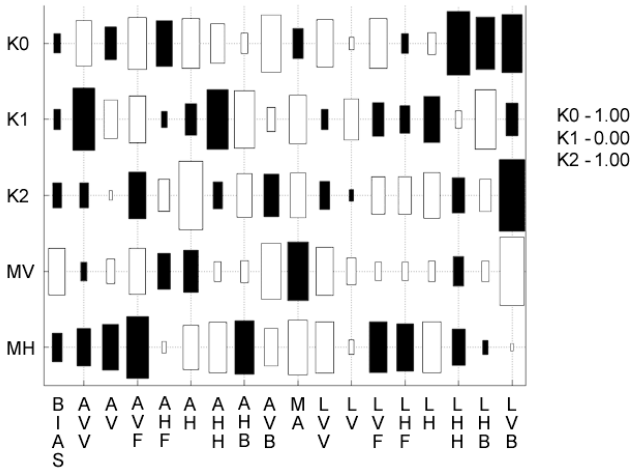



**Figur 17: Aktivationskurvor för noderna i agentens kontrollarkitekturen.**

Till skillnad från ett enkelt återkopplat nätverk kan den sensomotoriska mappningen i ett ESCN (mappningen mellan input och output) förändras under agentens interaktion med miljön. Detta gör att det krävs flera Hinton-diagram av den sensomotoriska mappningen för att förklara agentens beteende. Den sensomotoriska mappningen förändras dock enbart när beslutsnoden är aktiv.

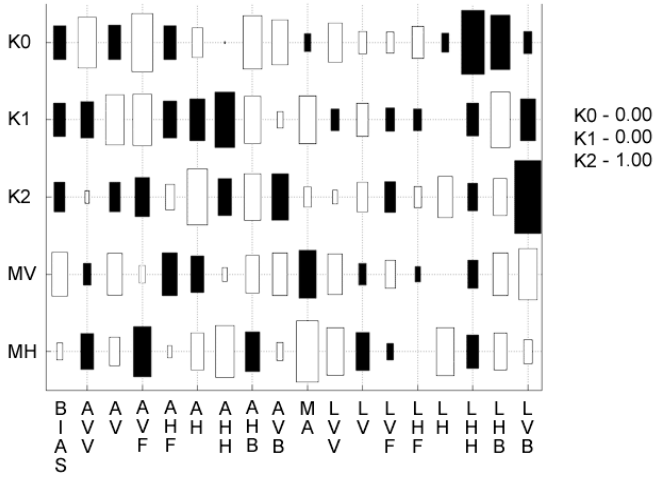
Beslutsnoden är aktiv under i stort sett hela uppgiften förutom när agenten svänger in i den första gången. Kontextnoderna påverkar alltså vikterna i funktionsnätverket vid nästan varje tidssteg. De nya vikterna är direkt beroende av kontextnodernas aktivation. Alltså förändras vikterna i nätverket enbart när aktivationen i kontextnoderna förändras. Samma aktivation i kontextnoderna vid två olika tillfällen ger samma vikter i nätverket följande tidssteg förutsatt att beslutsnoden är aktiv. I experimentet ger detta 5 unika sensomotoriska mappningar. Bytet mellan de sensomotoriska mappningarna under agentens färd mot målzonen beskrivs grafiskt i Figur 16.

När agenten startar följer kontrollarkitekturen sensomotorisk mappning 1, se Figur 18.




**Figur 18: Hinton-diagram över sensomotorisk mappning 1. Leder till ett ”följa den högra väggen-beteende”.**

Agenten står vänd i riktning mot det första ljuset i korridoren. Motornod MV aktiveras på grund av en positiv bias. Motornod MH är aktiverad så länge som agenten befinner sig nära en vägg på den högra sidan, på grund av en positiv koppling från avståndssensorerna AH och AHH. Tillsammans leder detta till att roboten följer den högra väggen. Avviker agenten från väggen avaktiveras avståndssensorerna AH och AHH (som sitter på agentens högra sida) vilket i sin tur leder till att även MH avaktiveras. Eftersom MV är aktiverad kommer detta att leda till att agenten roterar åt höger och ”söker” sig mot den högra väggen. Alltså leder sensomotorisk mappning 1 till ett ”följa den högra väggen-beteende”. När agenten når det första ljuset och LHH aktiveras avaktiveras kontextnod K0. Den sensomotoriska mappningen byts ut, från sensomotorisk mappning 1 till sensomotorisk mappning 2, se Figur 19.

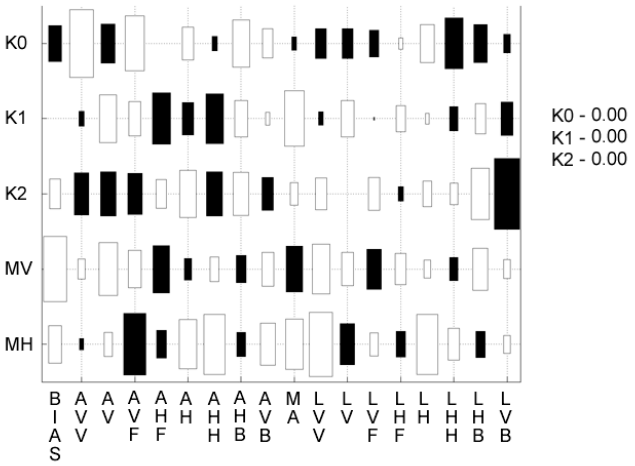



**Figur 19: Hinton-diagram över sensomotorisk mappning 2. Leder till ett ”följa den högra väggen-beteende”.**

I sensomotorisk mappning 2 har båda motornoderna positiv bias, dock är biasen för MH så pass liten att agenten fortsätter att ha ett ”följa den högra väggen-beteende”. När agenten passerat ljuset påverkar inte längre LHH kontextnod K0 och K0 aktiveras. Detta leder till att den sensomotoriska mappningen byts ut, från

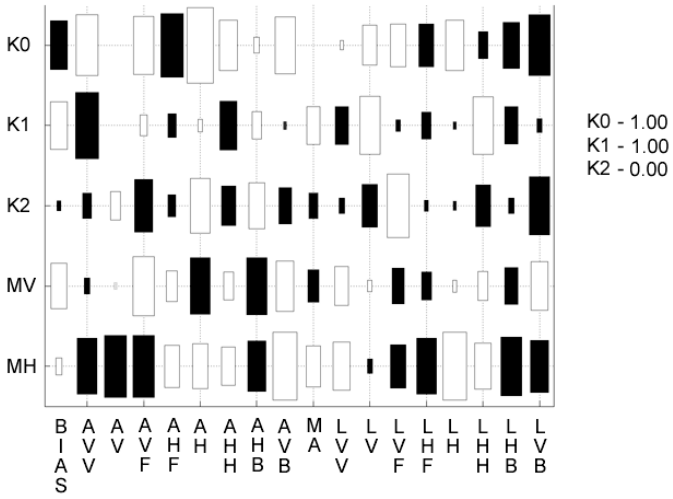
sensomotorisk mappning 2 till sensomotorisk mappning 1, se Figur 18. Agenten växlar mellan att följa sensomotorisk mappning 1 och 2 ända tills den passerat alla lampor.

När agenten passerar det första hörnet i den första gången försvinner aktivationen från AH och AHH. Den medelstarkt negativa biasen i MH leder till att aktivationen i MH sänks och agenten svänger åt höger. Avståndssensorerna AH och AHH påverkar inte längre K0 och K2 genom sina starkt positiva kopplingar. Den sensomotoriska mappningen byts ut, från sensomotorisk mappning 1 till sensomotorisk mappning 3, se Figur 20.



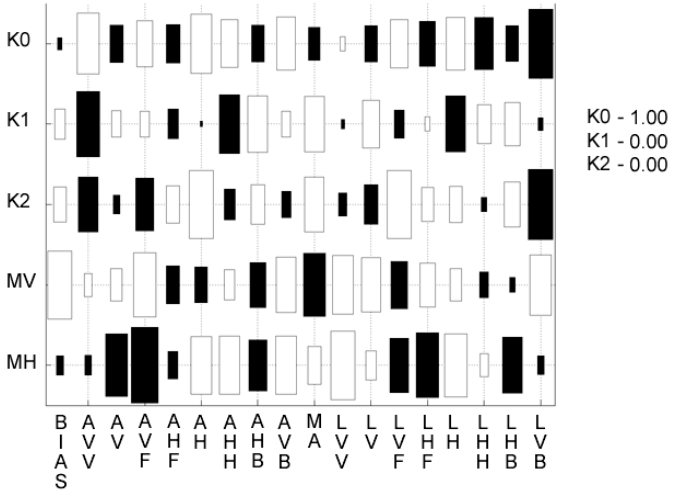

**Figur 20: Hinton-diagram över sensomotorisk mappning 3. Leder till ett ”gå rakt fram-beteende”.**

Liksom MH i sensomotorisk mappning 2 har MH i sensomotorisk mappning 3 en positiv bias. Detta leder till att noden aktiveras och att agentens färdväg rätas ut. Agenten förflyttar sig rakt fram mot gångens bortre vägg. I och med att aktivationen i K1 inte längre hålls nere av AH och AHH aktiveras K1. Däremot aktiveras inte beslutsnoden med följden att vikterna i nätverket inte förändras. När agenten når den bortre väggen roterar den åt höger med följden att aktivationen i kontextnoderna blir tillräckligt hög för att aktivera beslutsnoden. Den sensomotoriska mappningen byts ut, från sensomotorisk mappning 3 till sensomotorisk mappning 4, se Figur 21.




**Figur 21: Hinton-diagram över sensomotorisk mappning 4. Ett ”mellansteg” till en nya sensomotorisk mappning.**

Den 4:e sensomotoriska mappningen håller sig bara i ett tidssteg. Den sensomotoriska mappningen byts ut, från sensomotorisk mappning 4 till sensomotorisk mappning 5, se Figur 22.




**Figur 22: Hinton-diagram över sensomotorisk mappning 5. Leder till ett beteende där agenten söker sig till objekt på sin högra sida.**

Aktivationen i MH fortsätter att vara låg och agenten roterar åt höger. När agenten roterat tillräckligt många grader aktiveras MH. Agenten fortsätter följaktligen rakt fram mot motsatt vägg. Den sensomotoriska mappningen byts sedan ut, från sensomotorisk mappning 5 till sensomotorisk mappning 1, se Figur 18.

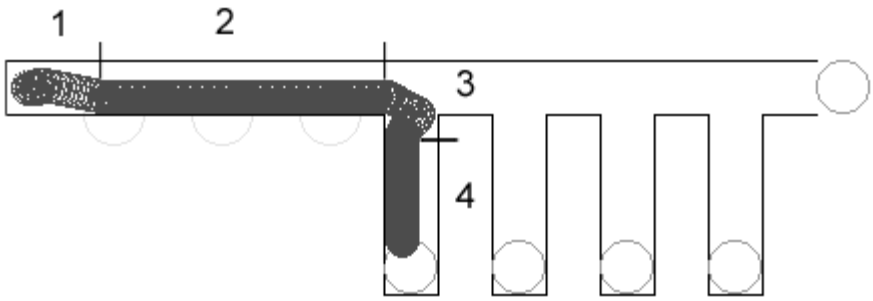
Med sensomotorisk mappning 1 upptar agenten återigen beteendet att följa den högra väggen, vilket leder agenten till (fel) målzon.

### 5.3.2 Enkelt återkopplat nätverk

I likhet med ett ESCN har agenten med ett enkelt återkopplat nätverk som kontrollarkitektur samma beteende oberoende av antalet lampor. Därför kommer det

även här enbart analyseras *en* situation att för att beskriva beteendet. Situationen som valts innehåller 3 lampor, vilket alltså är den mest komplexa situation agenten tränats på.

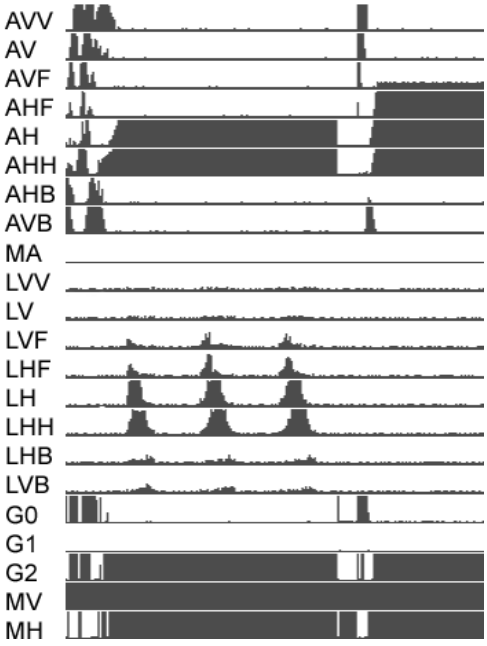
Agentens färdväg visas i Figur 23.




**Figur 23: Översiktsbild av agentens beteende i en situation med 3 lampor. Färdvägen har delats upp i sektioner för att underlätta beskrivningen av agentens interaktion med miljön.**

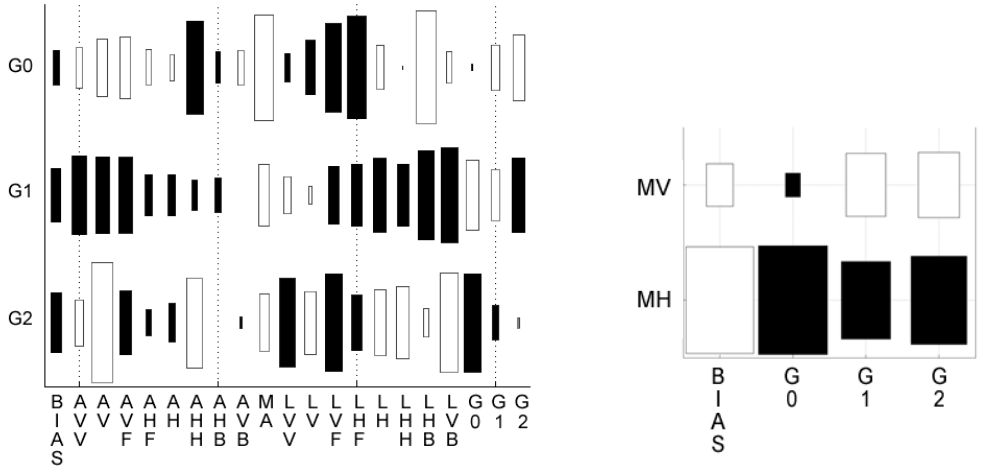
Översiktsbilden visar på att agenterna med de olika kontrollarkitekturerna har utvecklat ett i stort sett identiskt beteende. Skillnaden mellan agenten med ett enkelt återkopplat nätverk som kontrollarkitektur och agenten som har ett ESCN som kontrollarkitektur är att den sistnämnda förflyttar sig in i den första korridoren med en skarpare vinkel. Detta kompenseras sedan med en inte fullt så stor rotation när agenten når gångens bortre vägg. I övrigt visar båda agenterna på samma ”följa den högra väggen-beteende”.

Kontrollarkitekturens noder beskrivs med aktivationskurvor där en kurva motsvarar aktivationen i en nod under agentens interaktion med omgivningen, se Figur 24. Det är samma beteckningar som tidigare, dock med skillnaden att det finns 3 stycken gömda noder (G0, G1, G2) samt att det inte finns någon beslutsnod eller några kontextnoder.




**Figur 24: Aktivationskurvor för noderna i enkelt återkopplat nätverk.**

Agentens sensomotoriska mappning visas i två figurer. I Figur 25 visas (a) mappningen mellan inputnoder och gömda noder och (b) mappningen mellan gömda noder och outputnoder.




(a) (b)

**Figur 25: Hinton-diagram över vikterna för kopplingarna i enkelt återkopplat nätverk. (a) visar vikterna från inputnoder till gömda noder. (b) visar vikterna från gömda noder till outputnoder.**

När agenten startar är den vänd i svag vinkel mot korridorens vänstra vägg i förhållande till agenten, se sektion 1 i Figur 23. På grund av en positiv bias hos båda motornoderna färdas agenten framåt. Avståndssenorerna på den vänstra sidan (AVF och AV) aktiverar G0 genom en svag positiv koppling, med följden att även G2 aktiveras. Den gemensamma aktivationen i G0 och G2 avaktiverar motornod MH genom starkt negativa kopplingar och agenten roterar åt höger. I och med att aktivationen i AV försvinner avaktiveras G0 och G2 med följden att MH återigen aktiveras. Agenten förflyttar sig därför framåt.

När agenten närmar sig den högra väggen aktiveras AHH, se sektion 2 i Figur 23. Agenten fortsätter att färdas framåt i svag vinkel mot den högra väggen, med full aktivation i både MV och MH.

Agenten färdas hela vägen fram till den första gången, utan att påverkas av ljusstimuli. När den väl når den första gången försvinner aktivationen i AH och AHH, se sektion 3 i Figur 23. Motornod MH avaktiveras i något tidssteg och agenten roterar åt höger. Sedan aktiveras MH igen och agenten färdas följaktligen framåt mot den första gångens bortre vägg. När den når den bortre väggen aktiveras avståndssensorerna på agentens vänstra sida. Agenten upptar samma beteende som i sektion 1, det vill säga den söker sig bort från den vänstra väggen och följer den högra väggen till (fel) målzonen, se sektion 4 i Figur 23.

# 6 Diskussion

Studiens mål har varit att undersöka hur en agent med kontrollarkitekturen ”Extended Sequential Cascaded Network” (ESCN) hanterar en uppgift som kräver ett räkneliknande beteende. I testerna visade det sig att agenten hade samma beteende oberoende av antalet ljusstimuli. Eftersom agenten aldrig började att ”räkna” är det svårt att analysera om ett *räkneliknande beteende* baseras på traditionell representation, interaktionsbaserad representation eller något som överensstämmer med den dynamiska ansatsen. Skulle en analys utföras med denna utgångspunkt antas det innebära slutsatser som antingen baseras på andra studiers resultat eller som baseras på mindre viktiga iakttagelser i denna studie. Ett exempel på en sådan ”mindre viktig iakttagelse” är att agenten byter sensomotorisk mappning i samband med ljusstimuli. Den nya sensomotoriska mappningen är knappast en del av ett räkneliknande beteende vilket gör observationen oväsentlig med utgångspunkt från fokus för denna studie. Vidare är det antagligen få som skulle ställa sig frågan om och i så fall hur agenten ”räknade” ljusstimuli i miljön eftersom den ur ett distalt perspektiv inte visar några tecken på att ta hänsyn till dem. Därför görs ingen närmare analys av de interna tillstånden med utgångspunkt från traditionell representation, interaktionsbaserad representation eller den dynamiska ansatsen.

Inledningsvis kontrollerades implementationen genom att replikera den enklaste delen av Thieme (2002). Agenten utvecklade snabbt ett för uppgiften framgångsrikt beteende (efter 160 generationer), vilket antas validera implementationen.

För att kontrollera om agentens misslyckande berodde på de karakteristika hos ett ESCN gjordes ett kompletterande experiment med ett enkelt återkopplat nätverk. Inte heller i detta fall lyckades agenten utveckla ett framgångsrikt beteende, vilket tyder på att komplikationerna under ESCN utveckling åtminstone inte enbart beror på dess speciella arkitektur.

I det kompletterande experimentet varierades dessutom färre delar av miljön mellan varje epok. Med färre variationer antogs att agenten skulle ha större möjlighet till att utveckla ett räkneliknande beteende. Hade agenten lärt sig beteendet och det även fungerat utanför den specifika inlärningsmiljön, det vill säga i ett test med de ordinarie variationerna i miljön, vore tillämpandet av Jakobis (1997) ansats (se avsnitt 2.7) tveksamt. Eftersom agenten inte utvecklade ett räkneliknande beteende kan dock en sådan slutledning inte göras.

Istället faller fokus på självorganisering (se avsnitt 2.2.1) som utvecklingsmetod för komplexa beteenden. Självorganisering innebär att agenten automatiskt anpassar sitt beteende i interaktion med omvärlden och att designern i så liten grad som möjligt påverkar utvecklingen av beteendet. Genom att designern inte påverkar utvecklingen kan risken öka för att den hamnar i ett sorts lokalt optima, där den lösning som antas vara optimal enbart är det i det närmaste grannskapet. Möjligen kan en mer styrd fitnessfunktion som belönar ”dellösningar” röna större framgång. Då uppkommer frågan om *hur* styrd fitnessfunktionen skall vara. En mer styrd fitnessfunktion innebär att designern i högre grad påverkar agentens utveckling, vilket ökar risken för att det blir designerns bild av hur problemet skall lösas som vägleder utvecklingen snarare än agentens egen. Det beteende som agenten utvecklar blir i så fall enbart en efterapning av vad designern på en specifik nivå instruerat den att göra. Detta lämnar lite kvar för systemet självt att lösa (se avsnitt 2.2).

Jämförs studiens resultat med Wiles och Elmans (1995) experiment (se avsnitt 2.8.1) framhävs svårigheterna med att överföra osituerade lösningar till situerade domäner. I Wiles och Elman (1995) styrs utvecklingen av nätverket genom förberedd input, tolkad output och utvalda träningsexempel. Viktiga bitar av interaktionen med omgivningen har åsidosatts för att enbart fokusera på funktionen ”räkning”. Det som har skapats är ett nätverk som räknar, precis som det har instruerats att göra.

I detta experiment har den situerade agenten försökt att anpassa sig till en betydligt komplexare och mindre tillrättalagd domän än den som används i Wiles och Elman (1995). Utvecklingen av agenten har skett genom självorganisering där designers roll relativt sett är liten. Dessa bitar utgör en viktig del av den situerade ansatsen (se avsnitt 2.4) men det är möjligt att de har försvårat utvecklingen till den grad att beteendet uteblev. Detta återspeglas i Beers (2000) ifrågasättande av den situerade ansatsen möjligheter till att modellera högre kognitiva förmågor (se avsnitt 2.3). Även om ett ”räkneliknande beteende” inte är ekvivalent med ”räkning” som högre kognitiv förmåga är det möjligt det är så pass komplext att utvecklingen försvåras av ansatsens designeroberoende syn.

## 6.1 Fortsatta studier

I Thieme (2002) används en miljö med tydliga ”landmärken”, exempelvis väggen på motsatt sida av korsningen. Detta ökar möjligheten till att utveckla ett beteende som står i relation till dessa ”landmärken”, det vill säga att agenten svänger åt höger eller vänster beroende på i vilken vinkel den närmar sig den motsatta väggen i korsningen. Detta beteende kan antas bli komplexare om agentens uppgift gäller fler än två korsningar eftersom ”informationen” i en viss vinkel nödvändigtvis borde vara större. Agenten behöver trots allt ”hålla reda” på hur den skall svänga i fler korsningar. En intressant fortsättning vore alltså att utöka Thiemes (2002) experiment med fler korsningar. Detta antas ge ett beteende som sätter sin tillit till de interna tillstånden, exempelvis genom att utnyttja de sensomotoriska mappningar, i större grad än tidigare.

Fördelen med ett sådant experiment i jämförelse med experimentet i denna studie är att beteendet åtminstone delvis har utvecklats i ett tidigare arbete (Thieme, 2002). Om agenten inte utvecklar ett framgångsrikt beteende i uppgifter med fler än två korsningar finns större möjligheter till att få indikationer på vad som kan vara fel.

# Referenslista


Baddeley, A. (1999) *Essentials of Human Memory*. East Sussex, UK: Psychology Press Ltd, Publishers.

Bechtel, W. (1997) *Dynamics and Decomposition: Are they compatible?* Washington University in St. Louis. Tillgänglig på Internet: http://mechanism.uscd.edu/~bill/research/dynamics.html [hämtad: 2003-02-28].

Beer, R. D. (1995) A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1), sid 173-215.

Beer, R. D. (2000) Dynamical approaches to cognitive science. *Trends in Cognitive Sciences* 4(3), sid 91-99.

Brooks, R. A. (1991a) Intelligence without reason. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, sid 569-595.

Brooks, R. A. (1991b) Intelligence without representation. *Artificial Intelligence*¸ 47, sid 139-159.

Carlsson, J. & Ziemke, T. (2001) YAKS – Yet Another Khepera Simulator. I: Rückert, Sitte & Witkowski (Red.) *Autonomous Minirobots for Research and Entertainment – Proceedings of the 5th International Heinz Nixdorf Symposium*, sid 235-241. Paderborn, Germany: HNI – Verlagsschriftenreihe.

Dorffner, G. (1997) Radical connectionism – a neural bottom-up approach to AI. I: G. Dorffner (red.) *Neural Networks and a New Artificial Intelligence*, sid 93-132. London, UK: International Thomas Computer Press.

Dreyfus, H. L. (1997) From Micro-Worlds to Knowledge Representation: AI at an Impasse. I: J. Haugeland (red.) *Mind Design II*, sid 143-182. Cambridge, MA: MIT Press.

Elman, J. L. (1990) Finding Structure in Time. *Cognitive Science*, 14, sid 179-211.

Jakobi, N. (1997) Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, 6.

Meeden, L. A. (1996) An incremental approach to developing intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. Vol 26, nr 3, sid 474-485.

Mehrotra, K., Mohan C. K. & Ranka, S. (1997) *Elements of Artificial Neural Networks*. Cambridge, MA: MIT Press.

Mondada, R., Franzi, E. & Ienne, P. (1993) Mobile robot miniaturization: A tool for investigation in control algorithms . I: T. Y. Yoshikawa & F. Miyazaki (Red.) *Proceedings of the Third International Symposium on Experimental Robots*. Berlin: Springer-Verlag.

Newell, A. & Simon, H. A. (1997) Computer Science as Empirical Inquiry: Symbols and Search. I: J. Haugeland (red.) *Mind Design II*, sid 81-110. Cambridge, MA: MIT Pess.

Niklasson, L. & Bodén, M. (1997) Representing structure and structured representations in connectionist networks. I: A. Browne (red.) *Neural Network Perspectives on Cognition and Adaptive Robotics*, sid 20-50. IOP Press.

Nolfi, S. (1997) Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*, 5(3-4), sid 343-363.

Palmer, S. E. (1978) Fundamental Aspects of Cognitive Representation. I: E. Rosch och B. B. Lloyd (red.) *Cognition and Categorization*, sid 259-303. Hillsdale, NJ: Lawrence Erlbaum.

Searle, J. R. (1997) Minds, Brains, and Programs. I: J. Haugeland (red.) *Mind Design II*, sid 183-204. Cambridge, MA: MIT Press.

Thieme, M. (2002) *Intelligence without hesitation*. Master dissertation. Department of Computer Science, University of Skövde, Sweden.

Van Gelder, T. J. (1999) *Revisiting the Dynamical Hypothesis*. Preprint No. 2/99, University of Melbourne, Department of Philosophy.

Wiles, J. & Elman, J. (1995) Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. I: *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*. Cambridge, MA: MIT Press.

Ziemke, T. (1999) Remembering how to behave: recurrent neural networks for adaptive robot behavior. I: Medsker och Jain (red.) *Recurrent Neural Networks: Design and Applications*, sid 341-376. Boca Raton: CRC Press.

Ziemke, T. (2000) *Situated Neuro-Robotics and Interactive Cognition*. Doctoral dissertation, Department of Computer Science, University of Sheffield, UK.

Ziemke, T. & Thieme, M. (under tryckning) Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adaptive Behavior*, 10(3/4).