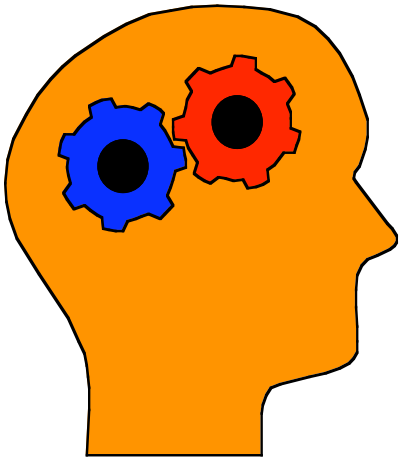
Ti forslag til datalogiske projekter inden for kunstig intelligens

Keld Helsgaun
RUC, august 1999



Kunstig intelligens er den gren af datalogien, som beskæftiger sig med automatisering af intelligent adfærd.

Området kan opdeles i to hovedgrene.

Den ene gren, kognitionsvidenskab, har en stærk tilknytning til psykologi. Her er målet at konstruere programmer til afprøvning af teorier, som beskriver og forklarer menneskelig intelligens.

Den anden gren er mere datalogisk orienteret og interesserer sig for, hvorledes datamater kan bringes til at udvise intelligent adfærd. Det er her ikke afgørende, om der er tale om efterligning af menneskets mentale processer, blot de konstruerede systemer udadtil reagerer intelligent.

I det følgende beskrives kort ti forslag til datalogiske projekter inden for den sidstnævnte gren. Forslagene retter sig primært mod studerende på datalogiuddannelsens første modul, hvor programmering indtager en central plads. Men forhåbentlig kan forslagene også være en inspirationskilde for uddannelsens øvrige studerende.

1. **Maskinindlæring** ved hjælp af beslutningstræer

Lige siden computeren blev opfundet, har der været stor interesse for at få den til at lære. Hvis vi kan programmere en computer til at *lære* - d.v.s. forbedre sin formåen på baggrund af sine erfaringer - vil det have vidtrækkende konsekvenser. Det vil f.eks. være muligt at få en computer til at ordinere en optimal behandling af en ny sygdom ud fra hidtidige erfaringer med behandling af en række beslægtede sygdomme. Muligheden for at få computere til at lære vil åbne for mange ny anvendelser af computere. Desuden kan man forestille sig, at viden om mulighederne for at automatisere læreprocesser vil give indsigt i menneskets evne til at lære.

Desværre ved vi endnu ikke, hvorledes vi kan får computere til at lære lige så godt som mennesker. Men de senere år er der fremkommet en del algoritmer, som med stor succes har muliggjort indlæring i forbindelse af visse typer af opgaver. For eksempel er de mest effektive algoritmer til talegenkendelse baseret på maskinindlæring.

Interessen for maskinindlæring er i dag så stor, at det er det mest aktive forskningsområde inden for kunstig intelligens.

Området kan opdeles i to dele, *symbolbaseret* og *ikke-symbolbaseret* maskinindlæring. Ved symbolbaseret indlæring er resultatet af læringen repræsenteret i form af symboler, enten i form af logiske udsagn eller grafstrukturer. Ved ikke-symbolbaseret indlæring er resultatet udtrykt som kvantitative størrelser, f.eks. som vægte i et *neuralt* netværk (en model af den menneskelige hjerne).

I de senere år har forskningen i neurale netværk været meget intensiv, og der er opnået bemærkelsesværdigt gode resultater. Det gælder f.eks. i forbindelse med tale- og billedgenkendelse.

Men også inden for symbolbaseret maskinindlæring er forskningen intensiv. En vigtig årsag til denne interesse er, at resultatet af en indlæring kan forklares og forstås af mennesker. Hvilket normalt ikke er tilfældet for neurale net. Dette forhold har stor betydning, hvis man skal kunne fæste lid til det indlærte.

Dette og det følgende projekt omhandler begge symbolbaseret maskinindlæring og begrænser sig til det område af feltet, der kaldes *induktion*.

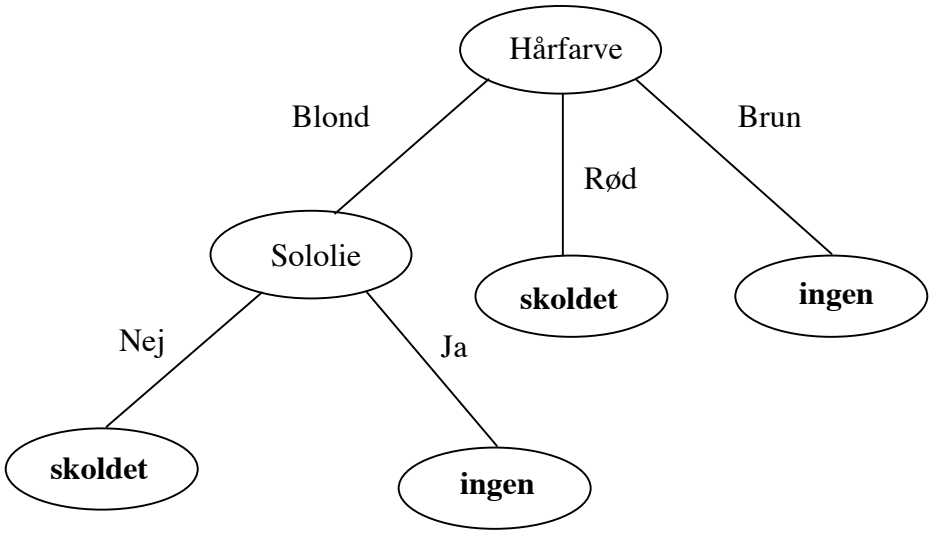
Logisk induktion består i at nå en generel konklusion ud fra en række eksempler. Et simpelt eksempel på induktion er følgende. Antag at du ser en række postkasser, der alle er røde. Ved induktion konkluderer du, at alle postkasser i hele verden er røde (også postkasser, du ikke har set).

Anvendelse af *beslutningstræer* er en af de mest anvendte metoder til indlæring ved logisk induktion. Brugen af et beslutningstræ illustreres bedst gennem et eksempel. Antag at du ikke kender de faktorer, der kan være årsag til at mennesker blive solskoldede. Du

observerer en række personer og registrerer nogle egenskaber ved dem, herunder om de bliver skoldet af solen. Observationerne er opført i nedenstående skema.

Navn	Hårfarve	Højde	Vægt	Sololie	Effekt
Sara	blond	middel	let	nej	skoldet
Dan	blond	høj	middel	ja	ingen
Alex	brun	lav	middel	ja	ingen
Anni	blond	lav	middel	nej	skoldet
Emilie	rød	middel	tung	nej	skoldet
Peter	brun	høj	tung	nej	ingen
John	brun	middel	tung	nej	ingen
Kate	blond	lav	let	ja	ingen

Ud fra dette skema kan vi konstruere følgende beslutningstræ.



Beslutningstræet kan bruges til at klassificere en given person (også en person, der ikke er med blandt de observerede personer). Start i toppen af træet og besvar spørgsmålene et efter et, indtil et blad nås. I bladet er klassifikationen lagret.

I det aktuelle tilfælde stemmer beslutningstræet overens med vores intuition omkring muligheden for at blive solskoldet. For eksempel spiller personens vægt og højde ingen rolle.

Det er faktisk muligt at konstruere mange beslutningstræer, der er i overensstemmelse med de observerede data. Men ikke alle er lige velegnede, når der skal foretages generaliseringer. Hvorledes konstrueres et træ, der er egnet til formålet?

Dette problem kan løses ved hjælp af den såkaldte *ID3-algoritme*, en af de mest effektive algoritmer til induktion. Algoritmen opbygger et træ, idet der tilstræbes et så simpelt træ

som muligt. Dette sker ud fra den teori, at et simpelt træ bedre kan klassificere fremtidige data end et mere komplekst træ. Algoritmen er baseret på den såkaldte *informationsteori*.

Nærværende projekt går ud på at implementere ID3-algoritmen og afprøve den på en række udvalgte eksempler. Hvis tiden tillader det, kan projektgruppen måske studere nogle forbedringer af den oprindelige algoritme, f.eks. C4.5-algoritmen. Eller gruppen kan sammenholde ID3-algoritmen med den såkaldte *versionsrum-algoritme*.

Litteratur:

[1] J. R. Quinlan:
Discovering rules by induction from large collection of examples,
i
D. Michie (editor),
Expert systems in the micro electronic age,
Edingburgh University Press (1979).

[2] J. R. Quinland:
Induction of decision trees,
Machine Learning, Vol. 1(1) (1986), pp. 81-106.

[3] J. R. Quinland:
C4.5: Programs for Machine Learning,
Morgan Kaufman (1993).

[4] T. M. Mitchell:
Machine Learning,
McGraw-Hill (1997),
Chapter 1-3, pp. 1-78.

[5] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Section 17.5, pp. 457-471.

[6] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Chapter 20-21, pp. 411-442.

[7] S. J. Russell & P. Norvig:
Artificial Intelligence - A Modern Approach,
Prentice Hall (1995),
Section 18.3-18.4, pp. 531-544.

[8] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Section 13.0-13.3, pp. 603-632.

[9] M. Ginsberg:
Essentials of Artificial Intelligence,
Morgan Kaufmann (1993),
Chapter 15, pp. 300-320.

[10] P. Clark & T. Niblett:
The CN2 Induction Algorithm.
Machine Learning, Vol. 3 (4), (1989), pp. 261-283.

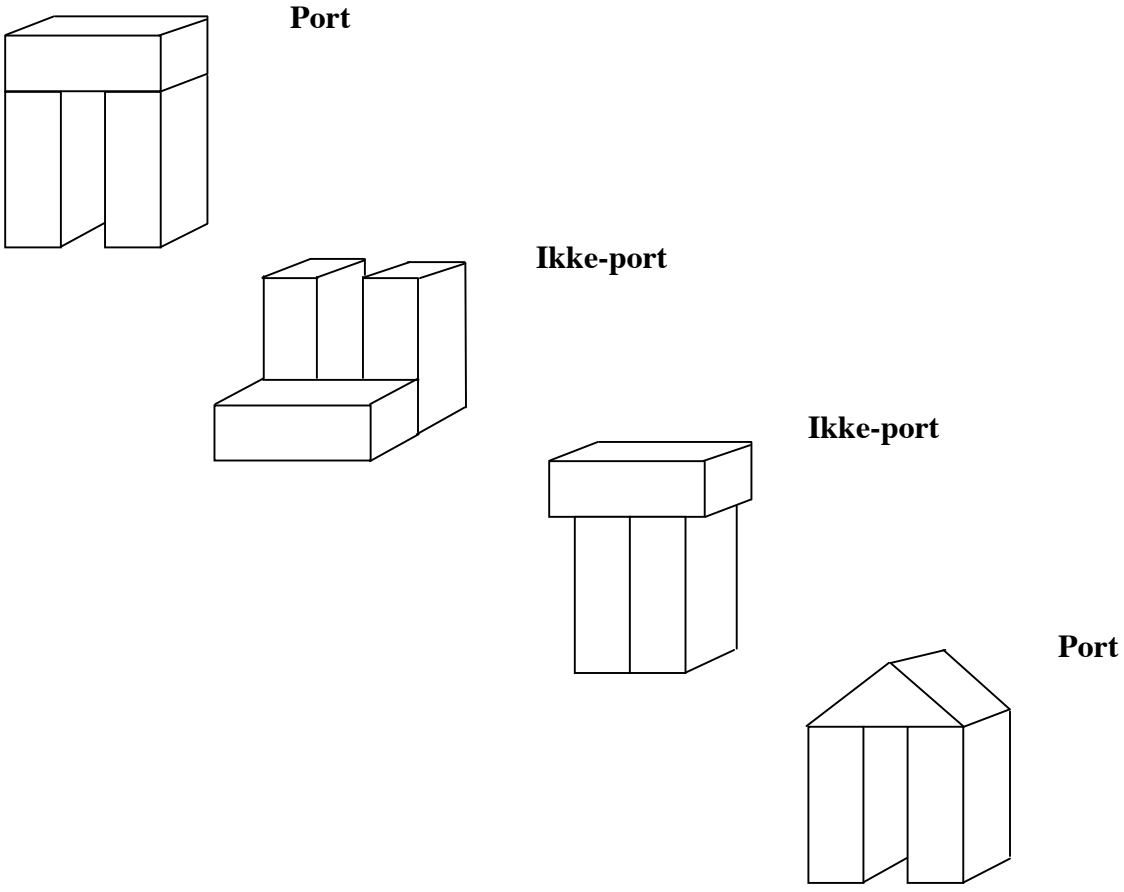
[11] *ID3kerne og Vindue – to generelle klasser til induktion ud fra ID3-algoritmen*,
RUC-rapport. Datalogi, modul 1 (1990/91).

2. Maskinindlæring ved analyse af forskelle

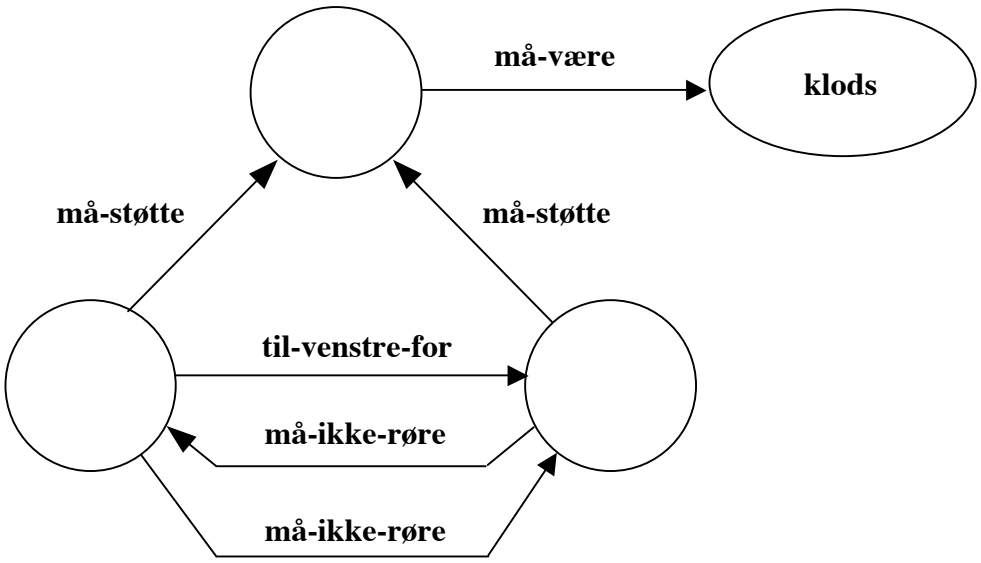
En af de første succesfulde algoritmer til maskinindlæring blev udviklet af Patrick Henry Winston i 1972. Algoritmen er i stand til at lære et begreb ved at analyse de forskelle, der optræder i en sekvens af observationer. For eksempel kan begrebet "port" indlæres på følgende måde.

Først præsenteres algoritmen for en række eksempler på opstillinger af klodser. Nogle af opstillingerne udgør en port. Disse kaldes for *positive eksempler*. Andre udgør ikke porte. De kaldes for *negative eksampler*.

Antag at algoritmen præsenteres for følgende eksempler.



Så vil algoritmen konkludere, at en port består af en klods, der understøttes af to kasseformede klodser, der ikke rører hinanden. Konklusionen repræsenteres i graf som den følgende (et såkaldt *semantisk net*).



Nærværende projekt går ud på at implementere Winstons originale algoritme. Algoritmen afprøves på et eller flere eksempler (bl.a. på ovennævnte port-indlæringsproblem).

Litteratur:

[1] P. H. Winston,
Learning structural descriptions from examples,
Ph.D. dissertation, MIT (1970).

[2] P. H. Winston,
Learning structural descriptions from examples,
i
Psychology of Computer Vision,
P. H. Winston (editor),
MIT Press (1975).

[3] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Chapter 16, pp. 349-363.

[4] R. S. Michalski,
Pattern Recognition as Rule-Guided Inductive Inference,
IEEE Transactions on Pattern Analysys and Machine Intelligence,
Vol. 2 (4), (1980).

[5] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Section 17.5.1, pp. 458-462.

[6] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Section 12.1, pp. 606-612.

[7] *Maskinindlæring. Winstons porte.*
RUC-rapport, Datalogi, speciale (1986/87).

3. Automatisk bevisførelse

Repræsentation og anvendelse af *viden* indtager en central plads i kunstig intelligens. Brug af viden er en forudsætning for intelligent adfærd.

Der er efterhånden udviklet en del formalismer til repræsentation af viden, hvoraf den mest anvendte er *logik*. Den logiske formalisme foretrækkes ofte, fordi den på en enkel måde gør det muligt at udlede ny viden ud fra eksisterende viden. Med denne formalisme kan det bevises, at et nyt udsagn er gyldigt, ved at påvise, at det følger logisk af de eksisterende, kendte udsagn.

Betragt følgende to udsagn.

> *Socrates er et menneske.*
> *Alle mennesker er dødelige.*

Ud fra udsagnene kan følgende nye udsagn konkluderes.

> *Socrates er dødelig.*

Logiske udsagn som ovenstående kan udtrykkes i den logiske formalisme, der kaldes *første-ordens prædikatlogik*. Udsagnene kan f.eks. udtrykkes således:

> menneske(Socrates).
> $\forall$x: menneske(x) $\Rightarrow$ dødelig(x)
> ---
> dødelig(Socrates)

Det andet udsagnene skal læses på følgende måde: For ethvert x gælder, at hvis x er et menneske, så er x dødelig.

Bevisførelsen foregår ved anvendelse af en eller flere slutningsregler. Bevisføerelsen i ovenstående eksempel er baseret på den slutningsregel, der kaldes *modus ponens*: ud fra udsagnene P og P $\Rightarrow$ Q udledes udsagnet Q.

Et gennembrud inden for automatisk bevisførelse skete i 1965, da J. A. Robinson [1] påviste, at ethvert bevis inden for første-ordens prædikatlogikken kan gennemføres ved brug af kun én slutningsregel, kaldet *resolution* (en generalisering af modus ponens).

Ved resolution bevises et udsagn ved hjælp af et *modbevis*. For at bevise et udsagn *negeres* udsagnet, og det forsøges påvist, at det negerede udsagn er i modstrid med de øvrige udsagn. Hvis det påvises, at det negerede udsagn er inkonsistent med de øvrige udsagn, så følger det, at det oprindelige udsagn må være gyldigt.

Det er i denne forbindelse værd at nævne, at resolution udgør grundlaget for det logiksprog, der kaldes PROLOG - i dag et af de mest anvendte sprog til programmering af systemer inden for kunstig intelligens.

Pladsen her tillader ikke en yderligere beskrivelse af første-ordens prædikatlogik og resolution. Der henvises til den anførte litteratur.

Nærværende projekt går ud på at implementere et program til automatisk bevisførelse. Programmet skal kunne indlæse en række logiske udsagn, konvertere dem til en intern form, kaldet *klausulform*, og kunne bevise, at et udsagn følger logisk af en mængde af andre udsagn. Bevisførelsen skal foretages ved hjælp af resolution.

Alt afhængig af projektgruppens ressourcer kan indlæsnings- og konverteringsdelen udelades. Boyer og Moores artikel [2] giver et godt udgangspunkt for implementeringen af programmets bevisdel.

Litteratur:

[1] J. A. Robinson:
A machine-oriented logic based on the resolution principle,
Journal of the ACM, Vol. 12 (1965), pp. 23-41.

[2] R. S. Boyer & J. S. Moore:
The sharing of structure in theorem-proving programs,
i
Machine Intelligence 7,
D. Mitichie, editor,
Elsevier (1972), pp. 101-116.

[3] J. A. Robinson,
The generalized resolution principle,
i
Machine Intelligence 3,
D. Mitichie, editor,
Elsevier (1968).

[4] R. S. Boyer & J. S. Moore:
Computational Logic,
Academic Press (1979).

[5] L. Wos,
Automated Reasoning,
Prentice-Hall (1988).

[6] N. J. Nilsson,
Problem-Solving Methods in Artificial Intelligence,
McGraw-Hill (1971).

[7] S. J. Russell & P. Norvig:
Artificial Intelligence - A Modern Approach,
Prentice Hall (1995),
Chapter 9, pp. 265-296.

[8] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Chapter 12, pp. 559-602.

[9] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Chapter 5, pp. 131-169.

[10] T. Østerby:
Kunstig intelligens - metoder og systemer,
Polyteknisk Forlag (1992),
Kapitel 2, s. 11-33,
Kapitel 7, s. 117-140.

[11] J. K. Siekmann & G. Wrightson (editors):
Automation of reasoning, Vol I and 2,
Springer Verlag (1983).

[12] M. Ginsberg:
Essentials of Artificial Intelligence,
Morgan Kaufmann (1993),
Chapter 6-8, pp. 105-172.

[13] R. J. Cunningham & S. Zappacost-Amboldi:
Software Tools for First-Order Logic,
Software - Practice and Experience,
Vol. 13 (1983), pp. 1019-1025.

[14] M. Fitting:
First-order logic and automated theorem proving,
Springer-Verlag (1990).

[15] *En SIMULA-klasse til automatisk bevisførelse*,
RUC-rapport. Datalogi, modul 3 (1977).

[16] Keld Helsgaun:
Automatisk bevisførelse,
DIKU-rapport. Speciale (1972/73).

4. Ekspertsystemer

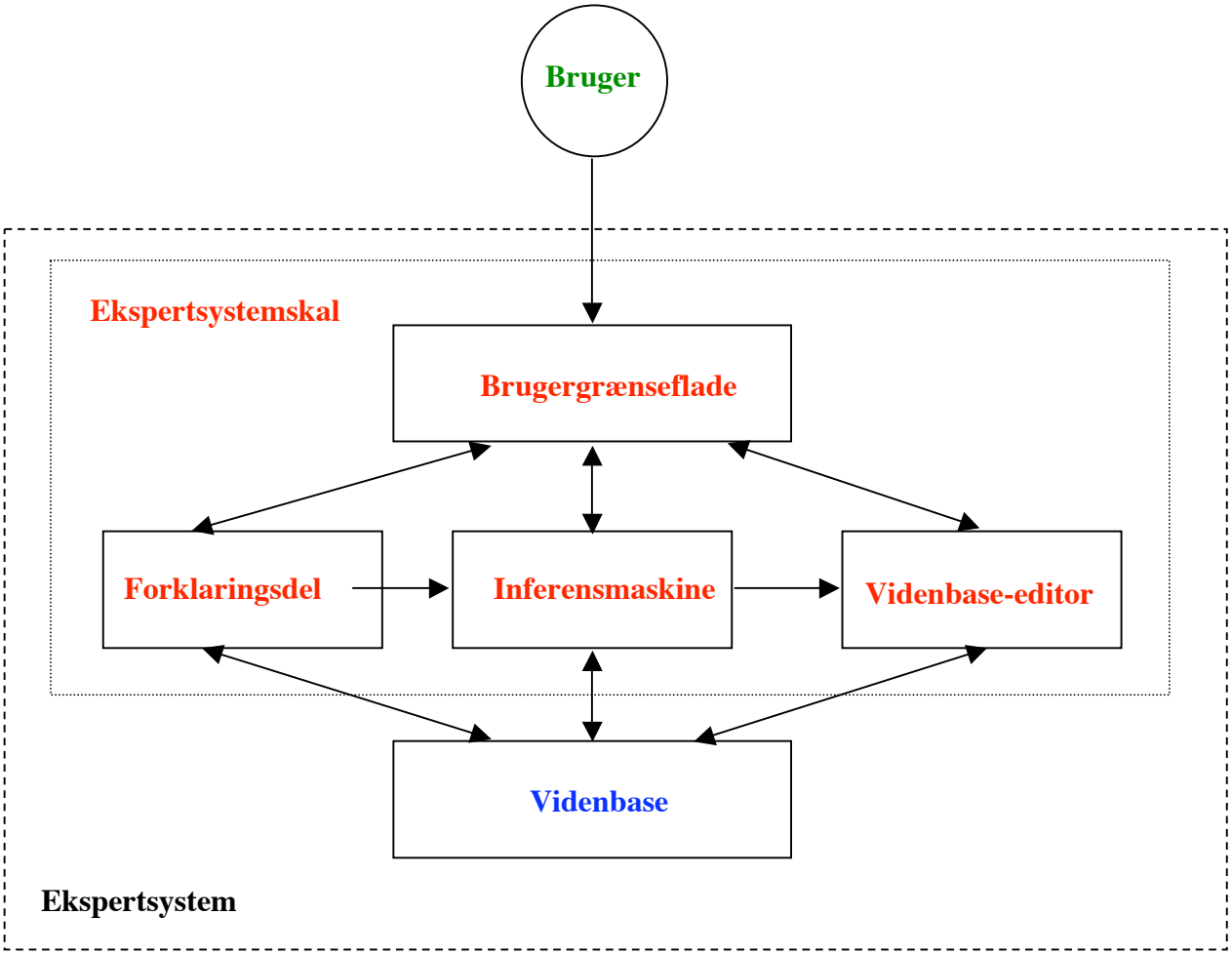
Menneskelige eksperter er i stand til at løse opgaver på et højt niveau, fordi de benytter stor viden om deres område. Dette er baggrunden for designet af programmer, der benytter sig af *vidensbaseret* problemløsning. Et *ekspertsystem*, som et sådan program ofte kaldes, anvender sig af specifik viden om et område for at opnå en kompetence, der kan sidestilles med en eksperts. Denne specifikke viden indhentes fra en eller flere eksperter inden for det pågældende område.

Ekspertsystemer er et af de områder inden for kunstig intelligens, der har haft størst anvendelsesmæssig succes. Ekspertsystemer anvendes i dag inden for en lang række fagområder, spændende fra medicin, kemi og geologi til jura, politik og økonomi. Ethvert område, hvor der skal træffes beslutninger, er et potentielt anvendelsesområde for ekspertsystemer.

Nedenstående liste angiver en række anvendelsesområder for ekspertsystemer [1].

1. Fortolkning:	dragning af konklusioner ud fra indsamlede data.
2. Forudsigelse:	beregning af konsekvenser i en given situation.
3. Diagnose:	bestemmelse af fejl ud fra observerede symptomer.
4. Design:	sammensætning af systemkomponenter, således at givne betingelser opfyldes.
5. Planlægning:	bestemmelse af en sekvens af handlinger, der fører til en ønsket situation.
6. Overvågning:	sammenligning af et systems adfærd med en forventet adfærd
7. Fejlreparation:	afhjælpning af fejl i et system.
8. Instruktion:	forbedring af indlært adfærd.
9. Kontrol:	styring af et systems adfærd.

Figuren på næste side viser de vigtigste dele af et ekspertsystem.



Videnbasen er den centrale del af et ekspertsystem. Den indeholder viden i form af fakta og regler. Sædvanligvis benyttes prædikatlogik til at beskrive den lagrede viden. Basen kan desuden indeholde information af, hvor *sikker* viden, der er tale om.

Inferensmaskinen benyttes til at drage logiske slutninger ud fra den lagrede viden.

Brugergrænsefladen benyttes til at udveksle information med systemets brugere, f.eks. til indlæsning af spørgsmål og til udskrivning af svar.

Forklararingsdelen giver brugeren information om systemets ræsonnering, herunder begrundelser for de opnåede svar.

Videnbase-editoren benyttes til at opbygge en videnbase og foretage ændringer i denne.

Adskillelsen af videnbasen fra de øvrige komponenter gør det muligt at genanvende disse øvrige komponenenter ved udviklingen af ekspertsystemer. En *ekspertsystemskal* indeholder alle programkomponeterne fra figuren ovenfor, pånær videnbasen. En udvikler af et ekspertsystem kan dermed benytte en "tom" skal og blot tilføje en relevant videnbase. Udvikleren kan dermed koncentrere sig om at indhente viden og opbygge videnbasen. Alle de øvrige komponenter i et ekspertsystem er nemlig umiddelbart tilgængelige.

I nærværende projekt implementeres og afprøves en ekspertsystemskal til udvikling af regelbaserede ekspertsystemer. Som udgangspunkt for implementeringen kan reference [2] benyttes med fordel.

Litteratur:

[1] D. A. Waterman:
A Guide to Expert Systems,
Addison-Wesley (1986).

[2] B. Sawyer & D. Foster:
Programming Expert Systems in Pascal,
Wiley Press (1986).

[3] T. Østerby:
Kunstig intelligens - metoder og systemer,
Polyteknisk Forlag (1992),
Kapitel 10, s. 179-198.

[4] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Chaper 20, pp. 547-557.

[5] S. J. Russell & P. Norvig:
Artificial Intelligence - A Modern Approach,
Prentice Hall (1995),
Chapter 5, pp. 122-148.

[6] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Chapter 6, pp. 207-246.

[7] M. Ginsberg:
Essentials of Artificial Intelligence,
Morgan Kaufmann (1993),
Chapter 18, pp. 381-401.

[8] F. Hayes-Roth, D. A. Waterman & D. B. Lenat (editors):
Building expert systems,
Addison-Wesley (1983).

[9] *En ekspertsystemskal i SIMULA*.
RUC-rapport. Datalogi, modul 1 (1995/96).

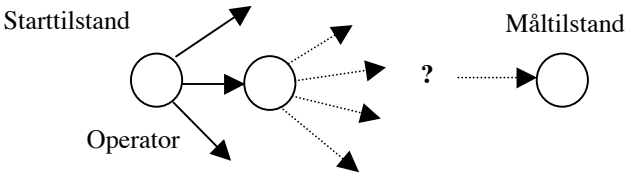
[10] *Nemesis - kernen i en skal. Ekspertsystemer og ekspertsystemskaller*,
RUC-rapport. Datalogi, speciale (1986/87).

5. Problemløsning ved hjælp af makro-operatorer

Problemløsning er et af de mest centrale emner i kunstig intelligens. Begrebet *problem* skal opfattes meget bredt og dækker enhver situation, hvor vi ønsker at finde ud af, om det er muligt at komme fra en given udgangsposition til en ønsket slutposition. Et problem defineres abstrakt ved følgende tre komponenter:

2. En *starttilstand*
3. En eller flere *måltilstande*
4. En mængde af *operatorer*

En operator fører fra en tilstand til ny tilstand. At løse problemet er ækvivalent med at finde en sekvens af operatorer, der fører fra starttilstanden til en måltilstanden.



Som et simpelt eksempel på et problem kan nævnes det såkaldte 8-problem.

Otte nummererede brikker er lagt på et kvadratisk bræt, f.eks. som vist nedenfor.

6	2	8
	3	5
4	7	1

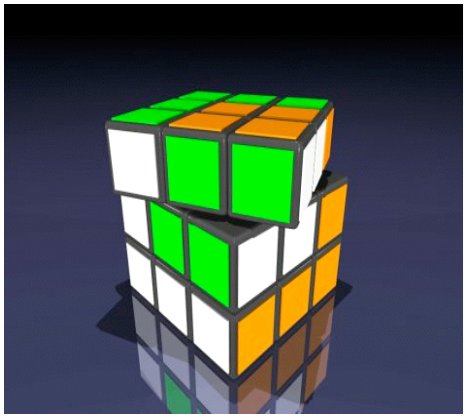
Et af felterne er tomt (det skraverede felt på figuren). I hvert træk er det muligt at flytte en brik til det tomme felt fra et af nabofelterne i samme række eller søjle.

Problemet er at finde en sekvens af lovlige træk, der fører fra starttilstanden til et angivet måltilstand, f.eks. nedenstående.

1	2	3
8		4
7	6	5

Dette og en lang række andre problemer kan løses ved hjælp af den såkaldte *A*-algoritme*. Lad der være givet et problem samt en *heuristisk* funktion, der for enhver tilstand bedømmer, hvor "tæt" tilstanden er på måltilstanden. En løsning af problemet bestemmes da ved en form for bedste-først-søgning. Algoritmen vælger i hvert skridt at arbejde videre med den tilstand, der aktuelt ser mest lovende ud med hensyn til at nå måltilstanden.

Imidlertid findes der også en del problemer, hvor A*-algoritmen ikke kan anvendes. Det gælder f.eks. for *Rubiks terning*.



Hver af terningens 6 sider er inddelt i 9 små felter, der hver har samme farve. Hver af de 6 flader kan drejes hver for sig. Hvis man nu skiftevis drejer på forskellige skiver, vil man konstatere at terningen hurtigt bliver broget. Problemet er nu at få drejet terningen tilbage til sin udgangsposition (som i dette tilfælde altså er måltilstanden).

Vanskeligheden ved at anvende A*-algoritmen til at løse dette problem består i, at der i dag ikke kendes nogen effektiv heuristik. Der er tilsyneladende ikke en simpel sammenhæng imellem terningens udseende og antallet af drejninger, der fører til måltilstanden. Selv om terningen ser temmelig broget ud, kan tilstanden godt være tæt på måltilstanden.

En metode til at løse problemet er at benytte sig af såkaldte *makro-operatorer*. En makro-operator er en fast sekvens af operatorer. I tilfældet med Rubiks terning er der tale om en fast sekvens af drejninger. Et problem løses ved at opdele det i en række uafhængige delproblemer, der så løses sekventielt. En makro-operator løser et givet delproblem uden at ødelægge løsningerne af de forudgående delproblemer.

Dette projekt går ud på at løse Rubiks terning ved hjælp af makro-operatorer, som beskrevet af R. E. Korf i publikationerne [1, 2, 3].

Litteratur:

[1] R. E. Korf:
Macro-opertors: A weak method of learning,
Artificial Intelligence,
Vol. 26, (1985), pp. 35-77.

[2] R. E. Korf:
Depth-first iterative deepening: An optimal admissable tree search,
Artificial Intelligence,
Vol. 17, (1985), pp. 97-109.

[3] R. E. Korf:
Leaning to Solve Problems by Searching for Macro-operators,
Ph.D. Thesis, Carnegie-Mellon University (1983).

[4] L. Kanal & V. Kumar, editors:
Search in artificial intelligence,
Springer-Verlag (1988).

[5] *Søgealgoritmer, makrooperatorer og kvadratspil*,
RUC-rapport, Datalogi, modul 1 (1997/98).

6. Problemløsning ved begrænsningsopfyldelse

Et *begrænsingsopfyldelses-problem*, i det følgende kaldet *CSP* (fra engelsk: Constraint Satisisfaction Problem), er en problemtype, der har følgende strukturelle egenskaber. Enhver problemtilstand er defineret ved et sæt af værdier af *variabler*, og måltilstanden opfylder visse *betingelser*.

Mere formelt kan et CSP beskrives på følgende måde. Et CSP består af

- en mængde af *variabler*, $\{x_1, x_2, \ldots, x_n\}$,
- for hver variabel, x_i, et *domæne*, D_i, af mulige værdier, og
- et sæt af betingelser, d.v.s. relationer imellem variablerne, der skal være opfyldt.

Opgaven går ud på for enhver variabel x_i at finde en værdi i D_i, således at alle betingelser er opfyldt.

Et simpelt eksempel på CSP er det såkaldte *8-dronninge-problem.* Problemet går ud på at placere 8 dronninger på et skakbræt, således at ingen dronning kan slå nogen anden dronning. Variablerne repræsenterer her placeringerne af hver af de 8 dronninger, mens betingelserne udtrykker, at to placerede dronninger ikke må kunne slå hinanden.

Et andet eksempel er følgende *kryptaritmetiske* problem.

```
  S E N D
+ M O R E
---------
M O N E Y
```

Hvilke forskellige cifferværdier fra 0 til 9 kan tildeles de enkelte bogstaver, hvis regnestykket skal stemme?

På grund af sin generalitet anvendes CSP i en lang række sammenhænge. Mange design- og planlægningsproblemer kan udtrykkes som CSPer. Det gælder f.eks. skemalægningsproblemer.

Dette projekt går ud på at implementere et generelt værktøj til løsning af CSP. Værktøjet afprøves på en række udvalgte eksempelproblemer.

Litteratur:

[1] A. K. Macworth:
Consistency in Networks of Relations,
Artificial Intelligence, Vol. 8 (1), (1977).

[2] E. C. Freuder:
Synthesizing Constraint Expressions,
Comminucations of the ACM, Vol. 21 (11), (1978).

[3] L. Kanal & V. Kumar, editors:
Search in artificial intelligence,
Springer-Verlag (1988).

[4] S. J. Russell & P. Norvig:
Artificial Intelligence - A Modern Approach,
Prentice Hall (1995),
Section 3.7, pp. 83-84.

[5] T. Østerby:
Kunstig intelligens - metoder og systemer,
Polyteknisk Forlag (1992),
Afsnit 6.8-6.9, s. 108-112.

[6] *KP - en generel klasse til Kombinatorisk Problemløsning*,
RUC-rapport. Datalogi, modul 1 (1990/91).

7. Spil

Spil har i tidens løb fascineret mange mennesker, og bestræbelser på at få computere til at spille spil har fundet sted mindst lige så længe, som der har eksisteret computere. Allerede i det attende århundrede overvejede Charles Babbage muligheden for at få en maskine til at spille skak, og i begyndelsen af 1950erne lavede Claude Shannon og Alan Turing de første udkast til programmer, der kunne spille skak.

Med udgangspunkt i Shannon og Turings ideer udviklede Arthur Samuel i begyndelsen af 1960erne et program, der kunne spille dam. Programmet var i stand til at lære af sine fejl og opnåede en imponerende styrke. Det var i stand til at vinde over den tids verdensmester.

Skak er et lang mere komplekst spil end dam, og først de senere år er det lykkedes at udvikle programmer, der kan vinde over de bedste menneskelige skakspillere.

Spil er et populært anvendelsesområde for kunstig intelligens. Der er flere årsager til dette, men den vigtigste er nok, at spil er velegnede til at evaluere nogle af de teknikker, der er centrale for kunstig intelligens, bl.a. søgning og brug af heuristisk viden. Et spil indeholder normalt relativt få og simple regler, og det er let at måle succes og fiasko.

Målet med dette projekt er udvikling af et program, der er i stand til at spille et givet spil. Spillet forudsættes at have følgende to egenskaber. Det skal være et spil, hvor to personer skiftes til at udføre et træk, et såkaldt *tomandsspil*, og det skal være et spil med *perfekt information*, d.v.s. et spil, hvor begge spillere kender den aktuelle stilling og de udførte træk. Blandt spil, som har disse egenskaber, kan nævnes dam, skak, Othello, Go, fem-på-stribe, backgammon og kalaha.

Som led i projektet udvikles en generel programpakke, der kan anvendes ved udvikling af program, der kan spille et givet tomandsspil. Programpakken skal tilbyde en søgemetode (f.eks. den såkaldte *Alpha-Beta-procedure*) til bestemmelse af et "godt" træk i en given stilling. Brugeren af programpakken skal give information om det aktuelle spil, herunder repræsentation af en stilling, lovlige træk og vurdering af en stilling.

Brugen af den udviklede programpakke demonstreres med et spil efter eget valg. Hvis gruppen har ressourcer til det, kan der eventuelt udvikles et grafisk brugergrænseflade for det pågældende spil.

Litteratur:

[1] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Chaper 12, pp. 307-326.

[2] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Chapter 6, pp. 101-118.

[3] S. J. Russell & P. Norvig:
Artificial Intelligence - A Modern Approach,
Prentice Hall (1995),
Chapter 5, pp. 122-148.

[4] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Section 4.3, pp. 144-152.

[5] T. Østerby:
Kunstig intelligens - metoder og systemer,
Polyteknisk Forlag (1992),
Kapitel 5, s. 77-91.

[6] D. E. Knuth & R. W. Moore:
An analysis of alpha-beta pruning,
Journal of Artificial Intelligence,
Vol. 6 (1975), pp. 18-24.

[6] *RUThello - en generel klasse i SIMULA til spilprogrammering. Og en implementering af spillet Othello.*
RUC-rapport. Datalogi, modul 1 (1996/97).

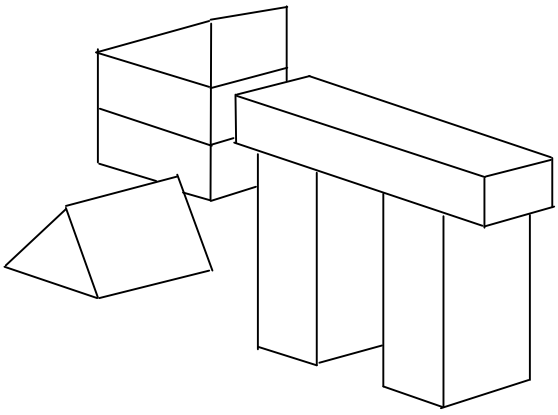
[7] *Et simula-program til damspil*,
RUC-rapport. Datalogi, modul 1 (1986/87).

[8] *Javaspil*,
RUC-rapprt, Datalogi, modul 1 (1998/99).

[9] *Tomandsspil*,
RUC-rapport. Datalogi, speciale (1985/86).

8. Datamatsyn

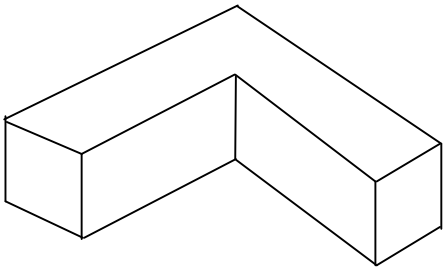
Betragt nedenstående tegning.



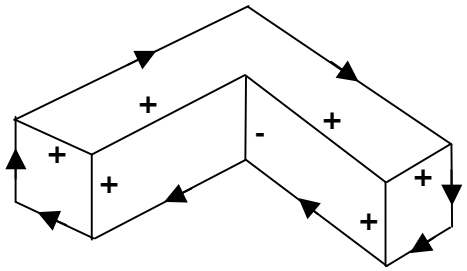
Vi mennesker har forholdsvis let ved at se, hvad tegningen forestiller. Vi kan for eksempel se, at der er flere objekter på figuren, og vi kan udpege dem. Anderledes forholder det sig med en computer. Ud fra kanterne på tegningen er det ikke enkelt at slutte sig til, hvilke objekter, tegningen indeholder.

At det kan lade sig gøre at udvikle en algoritme, der løser denne opgave, blev påvist af Waltz i 1975 [1]. Hans algoritme giver en fortolkning af hver kant på en tegning, som så derefter let kan bruges til at løse opgaven. For hver kant bestemmes, om kanten afgrænser et objekt, om det er en "ydre" (konveks) kant, eller om det er en "indre" (konkav) kant. En afgrænsende kant forsynes med en pil, således at hvis vi bevæger os i pilens retning, vil det pågældende objekt til højre. En ydre kant forsynes med et +, mens en indre kant forsynes med et -.

Antag for eksempel, at vi har følgende tegning af en klods.



Så vil resultatet af en fortolkning resultere, at kanterne forsynes med etiketter, som vist på nedenstående tegning.



Nærværende projekt går ud på at implementere og afprøve Waltz' algoritme. En letforståelig beskrivelse af algoritmen er givet i [2].

Litteratur:

[1] D. Waltz:
Understanding line drawings with shadows,
i
Computer Vision,
P. H. Winston (editor),
McGraw-Hill (1975), pp. 19-91.

[2] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Section 14.3, pp. 367-375.

[3] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Chapter 12, pp. 249-272.

[4] M. Ginsberg:
Essentials of Artificial Intelligence,
Morgan Kaufmann (1993),
Chapter 16, pp. 323-350.

[5] *Billedanalyse. Implementering af Waltz algoritme*,
RUC-rapport. Datalogi, modul 1 (1989/90).

9. Behandling af naturlige sprog

Evnen til at bruge et sprog til at kommunikere er måske den egenskab, der mest adskiller mennesker fra andre dyrearter. At forstå menneskelig tale er en vanskelig opgave, ikke mindst fordi den kræver behandling af analoge signaler. Disse signaler er normalt behæftet med støj. Men selv om opgaven simplificeres til forståelse af skeven tekst, er der tale om en meget vanskelig opgave. For at forstå skriftlige udsagn om et emne er det ikke blot nødvendigt at vide en masse om sproget selv (ordforråd, grammatik m.m.), men også en hel del om det pågældende emne.

Behandling af naturlige sprog er et område inden for kunstig intelligens, der blev startet i begyndelsen af tresserne. Området kan opdeles i følgende delområder:

- tekstforståelse
- tekstgenerering
- taleforståelse
- talegenerering
- maskinoversættelse (oversættelse fra et naturligt sprog til et andet)
- natursprogs-grænseflader til databaser

I de fleste anvendelser foretages en transformation af teksten eller det talte til en intern repræsentation. Transformation af sætninger til en intern repræsentation er derfor en central problemstilling.

Dette projekt går ud på at implementere grundelementerne i et system til forståelse af sætninger, der er skrevet i et naturligt sprog (f.eks. dansk eller engelsk).

Systemet tænkes realiseret gennem løsning af følgende delopgaver:

- morfologisk analyse
- syntaktisk analyse
- semantisk analyse

Den *morfologiske analyse* opdeler en sætning i ord. Hvert ord slås op i en ordbog for at bestemme ordets klasse og bøjningsformer. Disse oplysninger er inddata til den syntaktiske analyse.

Den *syntaktiske analyse* undersøger, om ordene kan sammenstilles til en tilladt sætning. Reglerne for hvilke sætninger, der er tilladt i sproget, udtrykkes i form af en *grammatik*. I projektet benyttes de såkaldte *udvidede tilstandsdiagrammer* til formålet. Disse kaldes også *ATN-grammatikker* (forkortelse af *A*ugmented *T*ransition *N*etworks). Resultatet fra den syntaktiske analyse er inddata til den semantiske analyse.

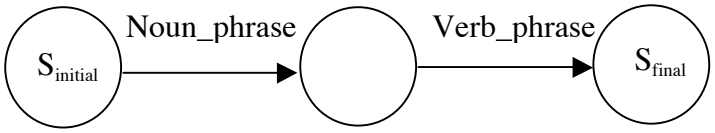
Den *semantiske analyse* bestemmer en sætnings mening, bl.a. ved at inddrage de enkelte ords mening i det pågældende sprog.

For at give et indtryk af metoden, skitseres i det følgende en analyse af sætningen

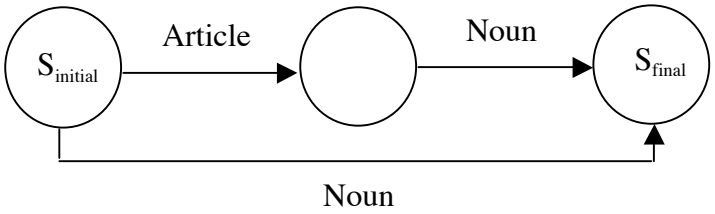
The dog likes a man.

Lad nedenstående tilstandsdiagram repræsentere grammatikken for en simpel delmængde af engelsk.

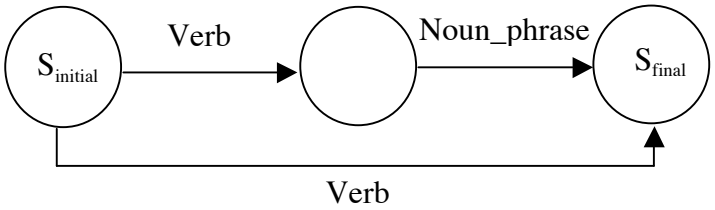
Sentence:



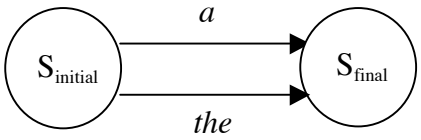
Noun_phrase:



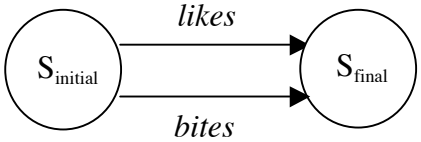
Verb_phrase:



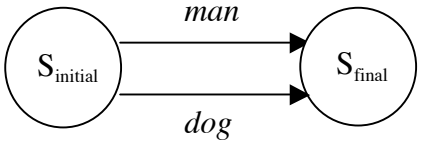
Article:



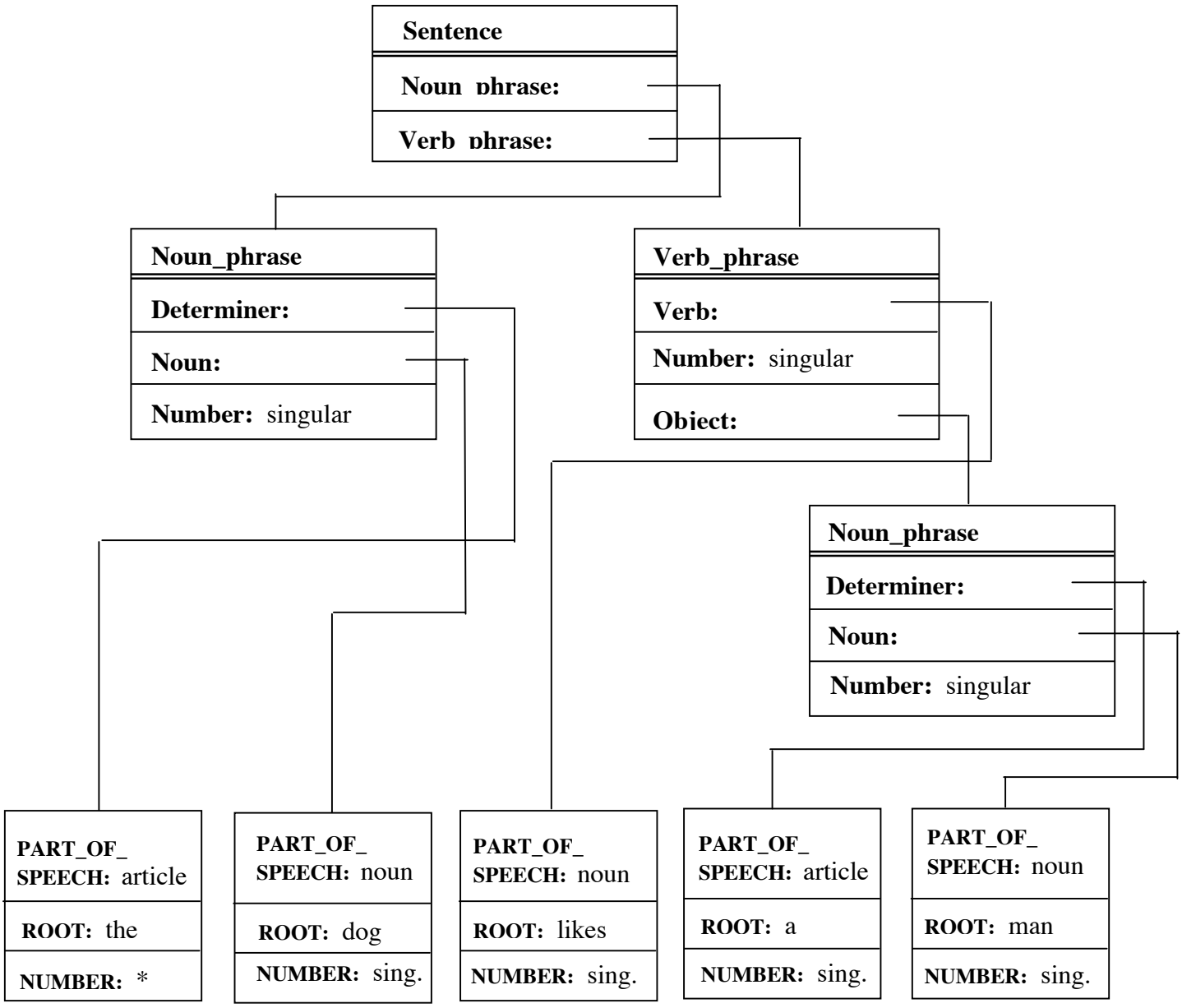
Verb:



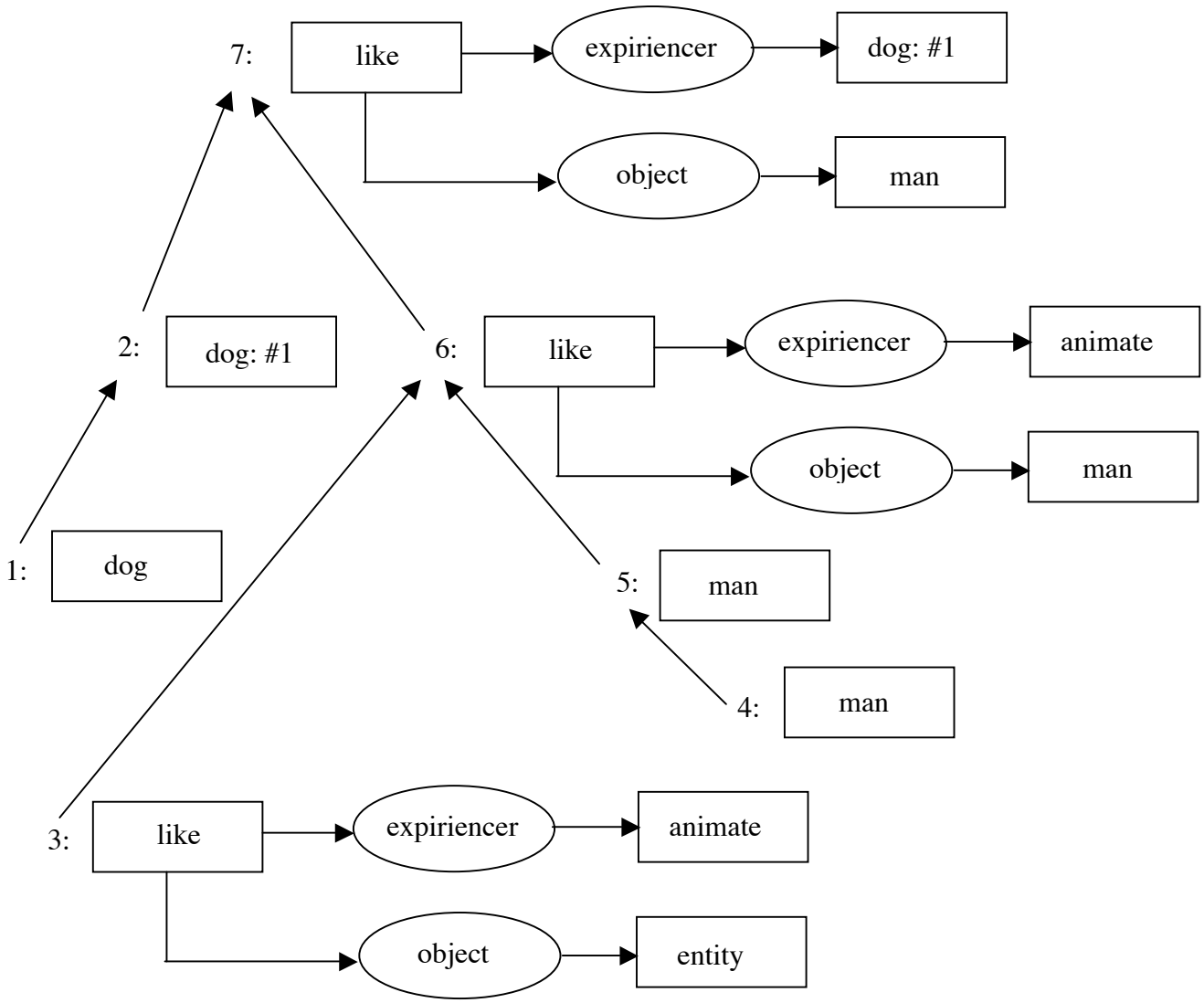
Noun:



Ved at udvide tilstandsdiagrammet med passende handlinger i diagrammets kanter, kan den oprindelige sætning transformeres til følgende *parsetræ*.



Dette parsetræ benyttes herefter som inddata for en semantisk analyse, hvilket resulterer i nedenstående semantiske repræsentation.



En nærmere forklaring til figurerne er givet i reference [1].

Litteratur:

[1] G. F. Luger &W. A. Stubblefield:
Artificial Intelligence - Structures and Strategies for Complex Problem Solving,
Addison-Wesley (1997),
Chapter 11, pp. 519-558.

[2] J. Allen:
Natural Language Understanding,
Benjamin/Cummings (2. edition), 1995.

[3] T. Winograd:
Language as a Cognitive Process,
Addison-Wesley (1983).

[4] M. D. Harris:
Introduction to Natural Language Processing,
Reston (1985).

[5] E. Rich & K. Knight:
Artificial Intelligence,
McGraw-Hill (1991),
Chapter 15, pp. 377-428.

[6] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Chapter 29, pp. 599-616.

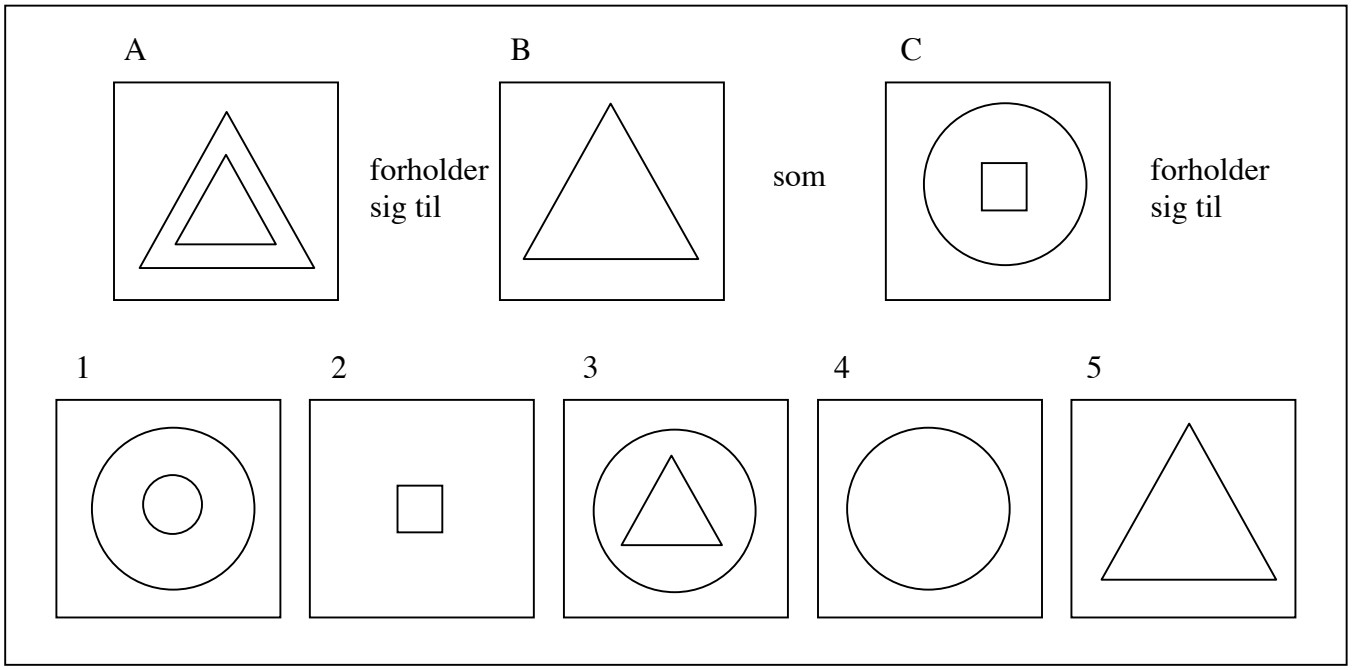
[7] M. Ginsberg:
Essentials of Artificial Intelligence,
Morgan Kaufmann (1993),
Section 7.2-7.3, pp. 354-371.

[8] T. Østerby:
Kunstig intelligens - metoder og systemer,
Polyteknisk Forlag (1992),
Kapitel 16, s. 275-303.

[9] *Natursprogsgrænseflade til SQL*,
RUC-rapport. Datalogi, modul 2 (1992/93).

10. Analogidannelse

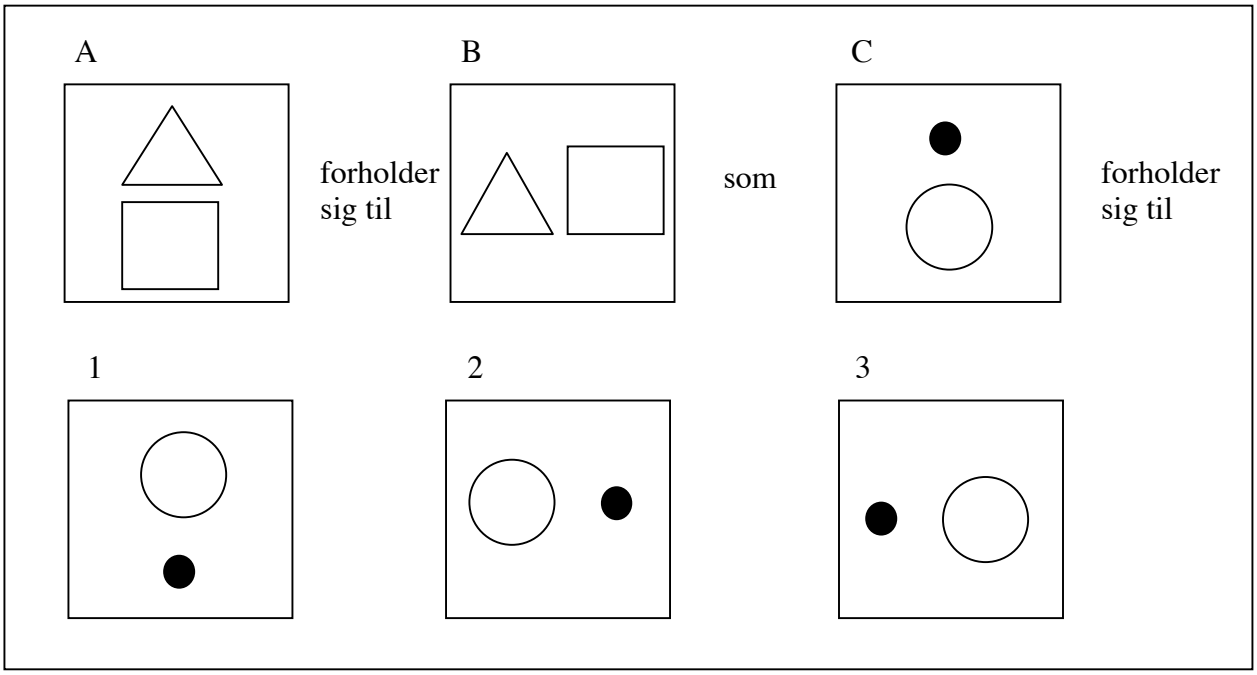
I intelligensprøver indgår ofte de såkaldte *geometriske analogiproblemer*. Forsøgspersonen præsenteres for en tegning som denne.



Opgaven går nu ud på at vælge en af figurerne 1 til 5, således at figur A forholder sig til figur B, som figur C forholder sig til den valgte figur.

Nærværende projekt går ud på at udvikle og afprøve et program, der kan løse den type opgaver. En mulig algoritme er beskrevet i referencerne [1] og [2]. Nedenfor skitseres ideen i algoritmen.

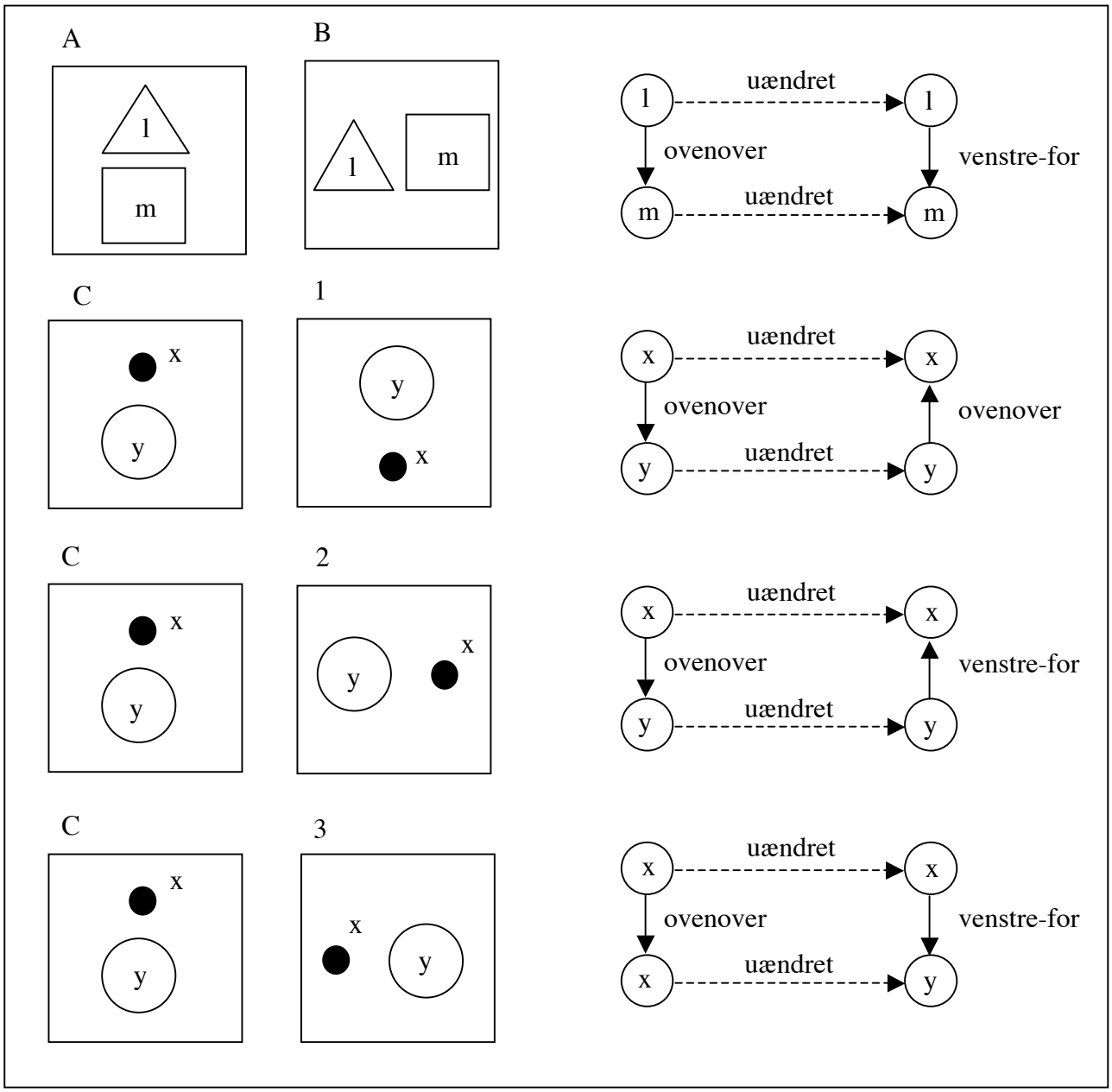
Antag at programmet præsenteres for følgende simple problem.



Så vil programmet opstille regler for, hvorledes A forholder sig til B, og for, hvorledes C forholder sig til hver af mulige svarfigurer. Den af reglerne for, hvorledes C forholder sig til et muligt svar, der bedst stemmer overens med reglen for, hvorledes A forholder sig til B, vil blive brugt til at identificere det bedste svar.

Hver regel består af to dele. Den ene del beskriver relationerne imellem de enkelte objekter i en figur. Et objekt kan være til venstre for, indeni eller ovenover et andet objekt. Den anden del beskriver, hvorledes objekterne i en figur transformeres til objekterne i en anden figur. Et objekt kan skaleres, roteres eller spejles, eller undergå en transformation, der er en kombination af disse operationer.

Regler beskrives ved hjælp af såkaldte *semantiske net*. For eksemplet er de resulterende semantiske net vist på næste side.



Ved sammenligning af de semantiske svarer reglen *C-forholder-sig-til-3* bedst til reglen *A-forholder-sig-til-B*. l og m skal blot erstattes med henholdsvis x og y. Det valgte svar bliver derfor figur nummer 3.

Litteratur:

[1] T. G. Evans:
A Heuristic Program to Solve Geometry Analogy Problems,
i
Semantic Information Processing,
M. Minsky (editor),
Academic Press (1968).

[2] P. Winston:
Artificial Intelligence,
Addison-Wesley (1992),
Section 2, pp. 15-45.