RAPPORTSERIE

Nr. 63b - Oslo

ESPEN REMMAN:

Data assistert konsekvensanalyse for økologien på Svalbard

NORSK POLARINSTITUTT

Nr. 63b - Oslo

ESPEN REMMAN:

Data assistert konsekvensanalyse for økologien på Svalbard

En hovedoppgave i informatikk,
studieretning databehandling,
ved Institutt for informatikk, Universitetet i Oslo

Espen Remman
Institutt for informatikk
Universitetet i Oslo
Postboks 1080 Blindern
0371 Oslo 3

ISBN 82-90307-64-0

DAKON - konsekvensanalyse av miljøforstyrrelser
på SVALBARD

Espen Remman

28. april 1990

Vedlegg C

Programmet

```
"World:Installasjon"
"*********************************************************"
"Innlesing av fakta om installasjonen. Det legges inn fakta om "
"sted, type virksomhet, tid, forurensing, ferdsel og forstyrrelse."
"*********************************************************"
"ai ->"
"Dette er hovedgoalet. Her starter analyse-prosessen. Det leses inn "
"navn, type, sted og tid for aktiv installasjon. I tillegg leses det"
"inn installasjons-avhengige attributter, forurensing, fedsel og "
"forstyrrelse. Fakta som allerede ligger i databasen blir ikke spurt etter! "
"En del fakta blir etterlyst avhengig av hva slags type(T) installasjon "
"det dreier seg om attributter(T)."

ai ->
   navn-ai
   type-ai(T)
   sted-ai(S)
   tid-ai
   attributter(T)
   ferdsel(F)
   forurensing(F-oru)
   forstyrrelse(F-orst);

"Legger opplysninger inn i databasen."

navn-ai -> ask(navnaiq,N);

type-ai(T) -> type-ask(typeaiq,virksomhet,T);

type(X,olje) -> eq(X,"oljeleting");
type(X,olje) -> eq(X,"oljeboring");
type(X,seismikk) -> eq(X,"marinseismikk");
type(X,seismikk) -> eq(X,"landseismikk");
type(X,feltarbeid) -> eq(X,"feltarbeid");
type(X,feltarbeid) -> eq(X,"feltarbeid");

"Kan her bruke is-a predikater"
"Spør om sted og tid"

sted-ai(S) -> sted(S);

"type-ask(stedaiq,sted,S);"

tid-ai ->
   type-ask(starttidaarq,tidaar,S-tartaar)
   type-ask(starttidmndq,tidmnd,S-tartmnd)
   type-ask(sluttidaarq,tidaar,S-luttaar)
   type-ask(sluttidmndq,tidmnd,S-luttmnd);

"Henter inn flere relevante opplysninger avhengig av type installasjon"
"Cut fordi det finnes bare en type."

attributter(T) -> eq(T,"feltarbeid") /;
attributter(T) -> eq(T,"landseismikk") / sprengnings-omfang;
attributter(T) -> eq(T,"marinseismikk") / sprengnings-omfang;
attributter(T) ->
   eq(T,"oljeleting")
   /
   terreng-type
   adkomst-veier
   person-antall
   transport
   montasje
   bore-utstyr;
attributter(T) ->
   eq(T,"oljeboring")
   /
   terreng-type
   adkomst-veier
   person-antall
   transport
   montasje
   bore-utstyr;

"Seismikk"

sprengnings-omfang -> antall-detoneringer kg-dynamitt km-produksjon;

antall-detoneringer ->
   ask(antdetonq,D);

kg-dynamitt ->
   ask(kgdynamittq,K-gD);

km-produksjon ->
   ask(kmprodq,K-mP);

"Olje"

terreng-type ->
   type-ask(typeterrengq,terreng,T);
```

```
adkomst-veier ->;

person-antall -> ask(antpersq,P);

transport ->
   type-transport
   tyngste-transport-enhet;

type-transport ->;

tyngste-transport-enhet ->;

montasje ->
   montasje-tidspunkt
   demonterings-tidspunkt;

bore-utstyr ->
   type-ask(typeboreutstyrq,boreutstyr,B);

montasje-tidspunkt ->;

demonterings-tidspunkt ->;

"************************************************************"
"************************************************************"
"          Forurensning"
"Hva slags forurensning"
"forurensning(F) ->"
"henter avhengig av type installasjon inn opplysninger om trafikk, "
"ferdsel og annen forurensende virksomhet på installasjonen. Disse "
"opplysningene prøves å samles til en felles størrelse <F>."
"<forurensning/1> tar for seg h.h.v.luft-, vann- og jordforurensinger."

forurensing(N-F) ->
   luft(F-oruL)
   vann(F-oruV)
   jord(F-oruJ)
   foru-f(F-oruL,F-oruV,F-oruJ,F)
   null-en("foru",F,N-F);

foru-f(F-oruL,F-oruV,F-oruJ,F) ->
   val(add(F-oruL,add(F-oruV,F-oruJ)),F);

"------------------------------------------------------------"

luft(F-oru) ->
   brennstoff(B-r)
   forbrenning-avfall(F-br)
   luft-f(B-r,F-br,F-oru);

vann(F-oru) ->
   borevaeske(B-V)
   boreslam(B-S)
   olje(O)
   vann-f(B-V,B-S,O,F-oru);

jord(F-oru) ->
   borevaeske(B-V)
   boreslam(B-S)
   avfall(A)
   smoereolje(S-O)
   jord-f(B-V,B-S,A,S-O,F-oru);

luft-f(B-r,F-br,F-oru) ->
   val(add(F-br,B-r),F-oru);

vann-f(B-Vaeske,B-Slam,O-lje,F-oru) ->
   val(add(B-Vaeske,add(B-Slam,O-lje)),F-oru);

jord-f(B-Vaeske,B-Slam,A-vfall,S-Olje,F-oru) ->
   val(add(B-Vaeske,add(B-Slam,add(A-vfall,S-Olje))),F-oru);

"------------------------------------------------------------"
"Pr døgn."

brennstoff(L-trBr) ->
   forbruk-diesel(L-trD)
   forbruk-bensin(L-trB)
   forbruk-olje(L-trO)
   brennstoff-f(L-trD,L-trB,L-trO,L-trBr);

brennstoff-f(L-trD,L-trB,L-trO,L-trBr) ->
   val(add(L-trD,add(L-trB,L-trO)),L-trBr);

"------------------------------------------------------------"

forbrenning-avfall(float(K-bkm)) ->
   ask(kbkmavfallq,S-Kbkm)
   string-integer(S-Kbkm,K-bkm);

borevaeske(float(L-tr)) ->
   ask(ltrborvskq,S-Ltr)
```

```
   string-integer(S-Ltr,L-tr);

boreslam(float(K-bkm)) ->
   ask(kbkborslamq,S-Kbkm)
   string-integer(S-Kbkm,K-bkm);

olje(float(L-tr)) ->
   ask(ltrspolq,S-Ltr)
   string-integer(S-Ltr,L-tr);

smoereolje(float(S-mO)) ->
   ask(ltrsmoq,S-SmO)
   string-integer(S-SmO,S-mO);

avfall(float(K-bkm)) ->
   ask(kbkavfq,S-Kbkm)
   string-integer(S-Kbkm,K-bkm);

forbruk-diesel(float(L-tr)) ->
   ask(ltrdieselq,S-Ltr)
   string-integer(S-Ltr,L-tr);

forbruk-bensin(float(L-tr)) ->
   ask(ltrbensinq,S-Ltr)
   string-integer(S-Ltr,L-tr);

forbruk-olje(float(L-tr)) ->
   ask(ltrolq,S-Ltr)
   string-integer(S-Ltr,L-tr);

"_____"
"************************************************************"
"************************************************************"
"Hva slags forstyrrelse"
"forstyrrelse(F) ->"
"henter inn opplysninger om sted og type forstyrrelse. Hvis ikke"
"opplysningene er tilstede spør programmet brukeren."

forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,"feltarbeid")
   /
   helikopter(H)
   baat-trafikk(B)
   tur(T-u)
   frst-felt(H,B,T-u,F)
   null-en("forstyrr",F,N-F);
forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,"marinseismikk")
   /
   baat-trafikk(B)
   frst-marin(B,F)
   null-en("forstyrr",F,N-F);
forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,bre-seismikk)
   /
   helikopter(H)
   baat-trafikk(B)
   tur(T-u)
   frst-bre(H,B,T-u,F)
   null-en("forstyrr",F,N-F);
forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,"tundraseismikk")
   /
   helikopter(H)
   baat-trafikk(B)
   tur(T-u)
   transport(T-r)
   frst-tundra(H,B,T-u,T-r,F)
   null-en("forstyrr",F,N-F);
forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,"oljeleting")
   /
   tur(T-u)
   frst-oljelt(T-u,F)
   null-en("forstyrr",F,N-F);
forstyrrelse(N-F) ->
   ask(typeaiq,T)
   eq(T,"oljeboring")
   /
   tur(T-u)
   frst-oljebor(T-u,F)
   null-en("forstyrr",F,N-F);

"Forstyrrelser eksplisitte til ferdsel."
"Det meste av forstyrrelser kan trekkes ut fra aktive installasjoner."
"Aggregerings funksjonene."
```

```
frst-felt(H-elikopter,B-aat,T-ur,F-orst) ->
   val(add(H-elikopter,add(B-aat,T-ur)),F-orst);

frst-marin(B-aat,F-orst) -> val(B-aat,F-orst);

frst-bre(H-elikopter,B-aat,T-ur,F-orst) ->
   val(add(H-elikopter,add(B-aat,T-ur)),F-orst);

frst-tundra(H-elikopter,B-aat,T-ur,T-ransport,F-orst) ->
   val(add(H,add(B,add(T-ur,T-ransport))),F-orst);

frst-oljelt(T-ur,F-orst) -> val(T-ur,F-orst);

frst-oljebor(T-ur,F-orst) -> val(T-ur,F-orst);

"************************************************************"
"************************************************************"
"Hva slags ferdsel"
"ferdsel(F) ->"
"henter avhengig av type installasjon inn opplysninger om trafikk og "
"ferdsel, og prøver å samle dette til en felles størrelse. Disse "
"dataene blir lagt ned i databasen."

ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"feltarbeid")
   /
   helikopter(H)
   baat-trafikk(B)
   tur(T-u)
   ferds-felt(H,B,T-u,F)
   null-en("ferd",F,N-F);
ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"marinseismikk")
   /
   baat-trafikk(B)
   ferds-marin(B,F)
   null-en("ferd",F,N-F);
ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"breseismikk")
   /
   helikopter(H)
   baat-trafikk(B)
   tur(Tu)
   ferds-bre(H,B,T-u,F)
   null-en("ferd",F,N-F);
ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"tundraseismikk")
   /
   helikopter(H)
   baat-trafikk(B)
   tur(T-u)
   transport(T-r)
   ferds-tundra(H,B,T-u,T-r,N-F)
   null-en("ferd",F);
ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"oljeleting")
   /
   tur(T-u)
   ferds-oljelt(T-u,F)
   null-en("ferd",F,N-F);
ferdsel(N-F) ->
   ask(typeaiq,T)
   eq(T,"oljeboring")
   /
   tur(T-u)
   ferds-oljebor(T-u,F)
   null-en("ferd",F,N-F);

"_____"

helikopter(M-engde) ->
   antall-flyginger(A)
   antall-km-flyg(K-m)
   rute-flyg(D-ep,D-est)
   heli-f(A,K-m,D-ep,D-est,M-engde);

"Mengde is A*Km"

heli-f(A-ntflyg,K-m,V1,V2,M-engde) ->
   val(mul(A-ntflyg,K-m),M-engde);

baat-trafikk(B) ->
   ant-baat-turer(A-bt)
   baat-type(T-ype)
   baat-stoerrelse(S-toerrelse)
   rute-baat(D-ep,D-est)
   baat-f(T-ype,S-toerrelse,A-bt,D-ep,D-est,B);
```

```
baat-f(T-ype,S-toerrelse,A-ntBaatTurer,D-ep,D-est,B) ->
   val(A-ntBaatTurer,B);

tur(T) ->
   ant-turgaaere(A-ntall)
   sted-tur(S-ted)
   tur-f(A-ntall,S-ted,T);

tur-f(A-ntall,S-ted,T) -> val(A-ntall,T);

"-------------------------------------------------------------------"

antall-flyginger(float(A-ntFlyg)) ->
   ask(antflygq,S-AntFlyg)
   string-integer(S-AntFlyg,A-ntFlyg);

antall-km-flyg(float(A-ntKm)) ->
   ask(kmflygq,S-AntKm)
   string-integer(S-AntKm,A-ntKm);

rute-flyg(D-ep,D-est) ->
   ask(flygfraq,D-ep)
   ask(flygtilq,D-est);

ant-baat-turer(float(A-Bt)) ->
   ask(antbaatturerq,S-ABt)
   string-integer(S-ABt,A-Bt);

baat-type(T-ype) ->
   ask(typebaatq,T-ype);

baat-stoerrelse(float(S-t)) ->
   ask(dwtq,S-St)
   string-integer(S-St,S-t);

rute-baat(D-ep,D-est) ->
   ask(baatrutefraq,D-ep)
   ask(baatrutetilq,D-est);

ant-turgaaere(float(A-nt)) ->
   ask(antturq,S-Ant)
   string-integer(S-Ant,A-nt);

sted-tur(S-ted) ->
   type-ask(stedturq,sted,S-ted);

"-------------------------------------------------------------------"
" Aggregeringsfunksjonene."

ferds-felt(H-elikopter,B-aat,T-ur,F) ->
   val(add(H-elikopter,add(B-aat,T-ur)),F);

ferds-marin(B-aat,F) ->
   val(B-aat,F);

ferds-bre(H-elikopter,B-aat,T-ur,F) ->
   val(add(H-elikopter,add(B-aat,T-ur)),F);

ferds-tundra(H-elikopter,B-aat,T-ur,T-ransport,F) ->
   val(add(H-elikopter,add(B-aat,T-ur)),F);

ferds-oljelt(T-ur,F) ->
   val(T-ur,F);

ferds-oljebor(T-ur,F) ->
   val(T-ur,F);

"-------------------------------------------------------------------"

;End world: Normal
```

```
"World:Rein"
"human,rev,runana,rein,kalvo,mort,pred,kond,syk,vandr,repro,tilgjb"
"runana."
"*********************************************************"
"Analyse opplegget."
"Denne verden inneholder alle komponentene til Svalbard-"
"reinen."
"*********************************************************"

run-analysis-rein ->
   wrt-if(fortsettq)
   outml("***********REIN***********")
   line
   line
   sted(S-t)
   wrt-rd(kjoennreinq,K-j)
   wrt-rd(alderreinq,A-ld)
   wrt-rd(tidssteppq,T)
   string-integer(A-ld,I-Ald)
   string-integer(T,I-T)
   rein(attr(S-t,K-j,I-Ald,I-T),A-ntall)
   line
   line
   outm("Alder: ")
   out(I-Ald)
   outm(",  Kjønn: ")
   outm(K-j)
   outm(",  Sted: ")
   outm(S-t)
   outm(",  Antall: ")
   out(A-ntall)
   line
   run-analysis-rein;
run-analysis-rein -> not(wrt-if(fortsettq)) /;

"-------------------------------------------------------------------"
"*************************************************"
"*************************************************"
"Startinput"
"rein(attr(V,simle,0,0),Antall)"
"inneholder antall dyr i tidsskritt <0>;"
"-------------------------------------------------------------------"
"                          REIN "
"-------------------------------------------------------------------"
"rein(Attributter,Antall) ->"
"gir avhengig av kjønn, alder og tid antall rein i neste tidsskritt;"
"Startverdi i tidsstepp 0 må legges i rein-db!!"

rein(attr(S-ted,K-joenn,A-lder,0),100) ->;
rein(attr(S-ted,"simle",0,T-PlusEn),A-ntallFoedte) ->
   outml("REINSIMLE")
   total-foedte(attr(S-ted,"simle",0,T-PlusEn),A-ntallFoedte)
   ass-rein-db(asked-komp(rein(attr(S-ted,"simle",0,T-PlusEn),A-ntallFoedte),
   A-ntallFoedte))
   outml("REINSIMLE<");
rein(attr(S-ted,"bukk",0,T-PlusEn),A-ntallFoedte) ->
   outml("REINBUKK")
   total-foedte(attr(S-ted,"bukk",0,T-PlusEn),A-ntallFoedte)
   ass-rein-db(asked-komp(rein(attr(S-ted,"bukk",0,T-PlusEn),A-ntallFoedte),
   A-ntallFoedte))
   outml("REINBUKK");
rein(attr(S-ted,"simle",18,T-PlusEn),+0.0) ->;
rein(attr(S-ted,"simle",A-lder,T-PlusEn),A-ntallRein) ->
   outml("REINSIMLE")
   val(inf(A-lder,18),1)
   val(sub(A-lder,1),A-ldMinEn)
   val(sub(T-PlusEn,1),T)
   ask-komp(rein(attr(S-ted,"simle",A-ldMinEn,T),A-nt),A-nt)
   mortalitet(attr(S-ted,"simle",A-lder,T-PlusEn),M-ort)
   ask-komp(beskatt-rein(attr(S-ted,"simle",A-lder,T-PlusEn),B-eskatt),B-eskatt)
   val(sub(A-nt,mul(A-nt,add(M-ort,B-eskatt))),A-ntallRein)
   ass-rein-db(asked-komp(rein(attr(S-ted,"simle",A-lder,T-PlusEn),A-ntallRein),
   A-ntallRein))
   outml("REINSIMLE");
rein(attr(S-ted,"bukk",13,T-PlusEn),+0.0) ->;
rein(attr(S-ted,"bukk",A-lder,T-PlusEn),A-ntallRein) ->
   outml("REINBUKK")
   val(inf(A-lder,13),1)
   val(sub(A-lder,1),A-ldMinEn)
   val(sub(T-PlusEn,1),T)
   ask-komp(rein(attr(S-ted,"bukk",A-ldMinEn,T),A-nt),A-nt)
   mortalitet(attr(S-ted,"bukk",A-lder,T-PlusEn),M-ort)
   ask-komp(beskatt-rein(attr(S-ted,"bukk",A-lder,T-PlusEn),B-eskatt),B-eskatt)
   val(sub(A-nt,mul(A-nt,add(M-ort,B-eskatt))),A-ntallRein)
   ass-rein-db(asked-komp(rein(attr(S-ted,"bukk",A-lder,T-PlusEn),A-ntallRein),
   A-ntallRein))
   outml("REINBUKK");

"-------------------------------------------------------------------"
"Rekursiv prosedyre; Regner ut totale antall fødte dyr;"
"-------------------------------------------------------------------"
"SIMLER og BUKKER"
```

```
total-foedte(attr(S-ted,"simle",17,T),A-ntSimler) ->
   reproduksjon(attr(S-ted,"simle",17,T),R-p)
   ask-komp(rein(attr(S-ted,"simle",17,T),R),R)
   val(div(mul(R-p,R),2),A-ntSimler);
total-foedte(attr(S-ted,"simle",A-ldersTeller,T),A-ntSimler) ->
   val(inf(A-ldersteller,17),1)
   val(add(A-ldersTeller,1),T-ellerPlusEn)
   reproduksjon(attr(S-ted,"simle",A-ldersTeller,T),R-p)
   ask-komp(rein(attr(S-ted,"simle",A-ldersTeller,T),R),R)
   total-foedte(attr(S-ted,"simle",T-ellerPlusEn,T),A-nt)
   val(add(div(mul(R-p,R),+2.0),A-nt),A-ntSimler);
total-foedte(attr(S-ted,"bukk",17,T),A-ntBukker) ->
   reproduksjon(attr(S-ted,"bukk",17,T),R-p)
   ask-komp(rein(attr(S-ted,"bukk",17,T),R),R)
   val(div(mul(R-p,R),+2.0),A-ntBukker);
total-foedte(attr(S-ted,"bukk",A-ldersTeller,T),A-ntBukker) ->
   val(add(A-ldersTeller,1),T-ellerPlusEn)
   reproduksjon(attr(S-ted,"bukk",A-ldersTeller,T),R-p)
   ask-komp(rein(attr(S-ted,"bukk",A-ldersTeller,T),R),R)
   total-foedte(attr(S-ted,"bukk",T-ellerPlusEn,T),A-nt)
   val(add(div(mul(R-p,R),+2.0),A-nt),A-ntBukker);

"Må få en tilfeldig fordeling av kjønnene!! Slik det er nå"
"er det 50/50 simler og bukker"
"_____"
"Opptelling av Simler og Bukker"
"_____"

antall-rein(attr(S-ted,K-j,A-ld,T),A-ntRein) ->
   akkumm-rein(attr(S-ted,"simle",1,T),A-ntSim)
   akkumm-rein(attr(S-ted,"bukk",1,T),A-ntBukk)
   val(add(A-ntSim,A-ntBukk),A-ntRein);

akkumm-rein(attr(S-ted,"simle",18,T),A) ->
   val(+0.0,A);
akkumm-rein(attr(S-ted,"simle",N,T),A) ->
   val(inf(N,18),1)
   ask-komp(rein(attr(S-ted,"simle",N,T),A-ntall),A-ntall)
   val(add(N,1),N-PlusEn)
   akkumm-rein(attr(S-ted,"simle",N-PlusEn,T),A1)
   val(add(A1,A-ntall),A);
akkumm-rein(attr(S-ted,"bukk",13,T),A) ->
   val(+0.0,A);
akkumm-rein(attr(S-ted,"bukk",N,T),A) ->
   val(inf(N,18),1)
   ask-komp(rein(attr(S-ted,"bukk",N,T),A-ntall),A-ntall)
   val(add(N,1),N-PlusEn)
   akkumm-rein(attr(S-ted,"bukk",N-PlusEn,T),A1)
   val(add(A1,A-ntall),A);

"##############################################"
"fungerer ikke foreløpig"

akkumm-rein-it(attr(S-ted,"simle",A-ld,T),A) ->
   assign(antall,+0.0)
   enum(U,18)
   val(sub(U,1),N)
   ask-komp(rein(attr(S-ted,"simle",N,T),A-ntall),A-ntall)
   eq(T-emp,antall)
   assign(antall,val(add(T-emp,A-ntall)))
   eq(A,antall);
akkumm-rein-it(attr(S-ted,"bukk",A-ld,T),A) ->
   assign(antall,+0.0)
   enum(U,14)
   val(sub(U,1),N)
   ask-komp(rein(attr(S-ted,"bukk",N,T),A-ntall),A-ntall)
   eq(T-emp,antall)
   assign(antall,val(add(T-emp,A-ntall)))
   eq(A,antall);

"fungerer ikke foreløpig"
"########################################"

beite-behov(A-ttributter,B-ehov) ->
   antall-rein(A-ttributter,A-ntRein)
   val(div(A-ntRein,+3.0),B-ehov);

beite-pr-ind(A-ttributter,B-eite) ->
   tilgj-beite(A-ttributter,<T-ilgjBeite,B-ehov>)
   val(div(B-ehov,T-ilgjBeite),B-eite);

"_____"

"*********************************************************"
"*********************************************************"
"Mennesket"

ant-menn(P) ->
   ask(antpersq,S)
   string-integer(S,P);
```

```
"Finnes i aktive installasjoner."

andel-snikskyttere(A-ndelSnik) ->
   ask(andelsnikskyttq,S-ProsSnik)
   str-pros-real(S-ProsSnik,A-ndelSnik);

"-------------------------------------------------------------"

"***********************************************************"
"***********************************************************"
"Polarrev"
"polarrev(N,T,A-ntall);"
"Antall inneholder antallet polarrev for aldersgruppe <N> og "
"i tid <T>;"

polarrev(attr(S-ted,K-joenn,0,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,1,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,2,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,3,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,4,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,5,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,6,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,7,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,8,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,9,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,10,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,11,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,12,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,13,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,14,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,15,0),+1.0e+2) ->;
polarrev(attr(S-ted,K-joenn,16,0),+1.0e+2) ->;

"antall-polarrev(T,AntallP) ->"
"regner ut på grunnlag av polarrev-bestanden i tid <T> antallet"
" polarrev;"

antall-polarrev(T,A-ntallP) ->
   akkumm-polarrev(attr(V1,V2,0,T),A-ntallP);

"-------------------------------------------------------------"
"akkumm-polarrev(attr(-,-,N,0),A) ->"
"Initielle (default) verdier se over;"
"akkumm-polarrev(Attributter,A) ->"
"går fra alder <N> opptil 16 som er høyeste alder på polarrev og "
"legger i A summen av polarrever i i alle aldersklasser;"
"akkumm-polarrev(Attributter,A) ->"
"vanlig utregning; Sjekker at vi ikke går ut over høyeste alder; "
"I tillegg unngår vi TID=0 siden denne ivaretaes av default "
"initieringen;"

akkumm-polarrev(attr(V1,V2,N,0),A) ->
   val(inf(N,17),1)
   polarrev(attr(V1,V2,N,0),A-ntall)
   val(add(N,1),N1)
   akkumm-polarrev(attr(V1,V2,N1,0),A1)
   val(add(A1,A-ntall),A);
akkumm-polarrev(attr(V1,V2,17,T),A) -> val(+0.0,A);
akkumm-polarrev(attr(V1,V2,N,T),A) ->
   val(inf(N,17),1)
   val(inf(0,T),1)
   ask-komp(polarrev(attr(V1,V2,N,T),A-ntall),A-ntall)
   val(add(N,1),N1)
   akkumm-polarrev(attr(V1,V2,N1,T),A1)
   val(add(A1,A-ntall),A);


"***********************************************************"
"***********************************************************"
"Ca normale mortaliteter hentet fra Svalbardreinen og dens livsgrunnlag;"

norm-mort(attr(V1,"simle",1,V2),+1.3e-1) ->;
norm-mort(attr(V1,"simle",2,V2),+6.0e-1) ->;
norm-mort(attr(V1,"simle",3,V2),+6.0e-1) ->;
norm-mort(attr(V1,"simle",4,V2),+1.6e-1) ->;
norm-mort(attr(V1,"simle",5,V2),+1.5e-1) ->;
norm-mort(attr(V1,"simle",6,V2),+1.4e-1) ->;
norm-mort(attr(V1,"simle",7,V2),+1.6e-1) ->;
norm-mort(attr(V1,"simle",8,V2),+1.8e-1) ->;
norm-mort(attr(V1,"simle",9,V2),+2.1e-1) ->;
norm-mort(attr(V1,"simle",10,V2),+2.4e-1) ->;
norm-mort(attr(V1,"simle",11,V2),+2.0e-1) ->;
norm-mort(attr(V1,"simle",12,V2),+3.5e-1) ->;
norm-mort(attr(V1,"simle",13,V2),+5.0e-1) ->;
norm-mort(attr(V1,"simle",14,V2),+2.6e-1) ->;
norm-mort(attr(V1,"simle",15,V2),+1.6e-1) ->;
norm-mort(attr(V1,"simle",16,V2),+2.0e-1) ->;
norm-mort(attr(V1,"simle",17,V2),+8.0e-1) ->;
norm-mort(attr(V1,"simle",17,V2),+1.0) ->;
norm-mort(attr(V1,"bukk",1,V2),+1.3e-1) ->;
norm-mort(attr(V1,"bukk",2,V2),+6.0e-1) ->;
norm-mort(attr(V1,"bukk",3,V2),+3.0e-1) ->;
```

```
norm-mort(attr(V1,"bukk",4,V2),+5.0e-1) ->;
norm-mort(attr(V1,"bukk",5,V2),+1.5e-1) ->;
norm-mort(attr(V1,"bukk",6,V2),+1.6e-1) ->;
norm-mort(attr(V1,"bukk",7,V2),+2.3e-1) ->;
norm-mort(attr(V1,"bukk",8,V2),+3.5e-1) ->;
norm-mort(attr(V1,"bukk",9,V2),+5.0e-1) ->;
norm-mort(attr(V1,"bukk",10,V2),+6.3e-1) ->;
norm-mort(attr(V1,"bukk",11,V2),+7.5e-1) ->;
norm-mort(attr(V1,"bukk",12,V2),+8.4e-1) ->;
norm-mort(attr(V1,"bukk",13,V2),+1.0) ->;

"_____________________________________________________________"
"                        MORTALITET"
"_____________________________________________________________"
"Henter inn en initiell mortalitet for tidsskritt 0;"
"Hvis brukeren starter med tidsskritt 0 gjelder default!!! Det samme er"
"tilfelle for kondisjon;"
"mortalitet(Attributter,M) ->"
"Attributter=attr(Sted,Kjoenn,Alder,Tid);"
"beregner mortalitetsrate <M> på grunnlag av predasjon, kondisjon og"
"sykdom; Det taes utgangspunkt i en normal mortalitet <norm-mort/2>;"

mortalitet(attr(S-ted,K-joenn,A-lder,0),M-ort) ->
   outml("MORT")
   norm-mort(attr(V1,K-joenn,A-lder,V2),M-ort)
   outml("MORT<");
mortalitet(A-ttributter,N-M) ->
   outml("MORT")
   predasjon(A-ttributter,P)
   kondisjon(A-ttributter,V-ekt)
   kondis-mort(A-ttributter,V-ekt,M-ort)
   ask-komp(syk(A-ttributter,S),S)
   bez("Syk","Mort",syk-mort-bez(S,S-M))
   norm-mort(A-ttributter,N-ormM)
   mort-f(P,M-ort,S-M,N-ormM,M)
   null-en("mort",M,N-M)
   ass-rein-db(asked-komp(mortalitet(A-ttributter,N-M),N-M))
   outml("MORT<");

"syk-mort(S,S-ykMort) må legges inn!!!"
"################################################"
"            MORTALITETS-FUNKSJONEN"
"################################################"

mort-f(P-red,M-ort,S-yk,N-ormMort,N-yMort) ->
   val(add(P-red,add(M-ort,add(S-yk,N-ormMort))),N-yMort);

"################################################"
" kondis-mort  kondisjonens virkning på MORTALITETEN;"
"Kondisjonens innvirkning p} mortalitet;"
"Proporsjonal sammenheng mellom mortalitet og kondisjon!"

kondis-mort(A-ttributter,V-ekt,M-ort) ->
   vekt-rel-n-vekt(A-ttributter,V-ekt,R-ateNVekt)
   val(inf(R-ateNVekt,+6.0e-1),1)
   val(+1.0,M-ort);
kondis-mort(A-ttributter,V-ekt,N-Mort) ->
   vekt-rel-n-vekt(A-ttributter,V-ekt,R-ateNVekt)
   val(inf(+1.0,R-ateNVekt),0)
   km(R-ateNVekt,M-ort)
   null-en("kondis-mort",M-ort,N-Mort);

km(R-ateNVekt,M-ort) ->
   val(sub(+1.0,mul(+2.5,sub(R-ateNVekt,+6.0e-1))),M-ort);

kond-til-m(G-radKondisjon) ->
   ask(kondmortq,S-ProsKondisjon)
   str-pros-real(S-ProsKondisjon,G-radKondisjon);

"Sykdoms innvirkning p} mortaliteten;"

syk-til-m(G-radSyk) ->
   ask(sykmortq,S-GradSyk)
   str-pros-real(S-Gradsyk,G-radSyk);

"_____________________________________________________________"

"************************************************************"
"************************************************************"
"predasjon(attr(-,Kjoenn,Alder,0),0);"
"henter inn initiell predasjon som settes til <0> i tidsskritt <0>;"
"predasjon(Attributter,P) ->"
"Attributter=attr(Sted,Kjoenn,Alder,Tid);"
"beregner predasjon <P> i neste tidsskritt <Tid> for dyra med <Kjoenn> og"
"<Alder>;"
" ask-komp(rein(A-ttributter,A-ntallrein),A-ntallRein) "
"skal kanskje v re med"

predasjon(attr(S-ted,K-joenn,A-lder,0),+0.0) -> outml("PRED");
predasjon(A-ttributter,N-P) ->
   outml("PRED")
   eq(A-ttributter,attr(S-ted,K-joenn,A-lder,T-PlusEn))
```

```
    val(sub(T-PlusEn,1),T)
    ask-komp(polarrev-til-pr(A-ttributter,G-radP),G-radP)
    antall-polarrev(T,A-ntP)
    andel-snikskyttere(A-ndSnik)
    ant-menn(A-ntM)
    ask-komp(beskatt-rein(A-ttributter,B-eskattAndel),B-eskattAndel)
    pred-f(G-radP,float(A-ntP),A-ndSnik,float(A-ntM),B-eskattAndel,P)
    null-en("pred",P,N-P)
    ass-rein-db(asked-komp(predasjon(A-ttributter,N-P),N-P))
    outml("PRED<");

"################################################"
"            PREDASJONS-FUNKSJONEN"
"################################################"

pred-f(G-radP,A-ntP,A-ndSnik,A-ntM,B-eskattAndel,P) ->
    val(add(add(mul(G-radP,A-ntP),mul(A-ndSnik,A-ntM)),B-eskattAndel),P);

"################################################"
" Vi finner antall dyr skutt ved snikskyting ved å multiplisere andel dyr "
"ulovlig felt pr. menneske  med antall menesker i området;"
"-----------------------------------------------------------------------"

"*********************************************************"
"*********************************************************"

ideell-vekt(attr(V1,"simle",0,V2),+1.2e+1) ->;
ideell-vekt(attr(V1,"simle",1,V2),+2.0e+1) ->;
ideell-vekt(attr(V1,"simle",2,V2),+3.0e+1) ->;
ideell-vekt(attr(V1,"simle",3,V2),+3.7e+1) ->;
ideell-vekt(attr(V1,"simle",4,V2),+4.3e+1) ->;
ideell-vekt(attr(V1,"simle",5,V2),+4.7e+1) ->;
ideell-vekt(attr(V1,"simle",6,V2),+5.0e+1) ->;
ideell-vekt(attr(V1,"simle",7,V2),+5.2e+1) ->;
ideell-vekt(attr(V1,"simle",8,V2),+5.2e+1) ->;
ideell-vekt(attr(V1,"simle",9,V2),+5.2e+1) ->;
ideell-vekt(attr(V1,"simle",10,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",11,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",12,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",13,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",14,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",15,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",16,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"simle",17,V2),+5.3e+1) ->;
ideell-vekt(attr(V1,"bukk",0,V2),+1.2e+1) ->;
ideell-vekt(attr(V1,"bukk",1,V2),+2.0e+1) ->;
ideell-vekt(attr(V1,"bukk",2,V2),+3.0e+1) ->;
ideell-vekt(attr(V1,"bukk",3,V2),+3.5e+1) ->;
ideell-vekt(attr(V1,"bukk",4,V2),+4.0e+1) ->;
ideell-vekt(attr(V1,"bukk",5,V2),+4.2e+1) ->;
ideell-vekt(attr(V1,"bukk",6,V2),+4.4e+1) ->;
ideell-vekt(attr(V1,"bukk",7,V2),+4.5e+1) ->;
ideell-vekt(attr(V1,"bukk",8,V2),+4.6e+1) ->;
ideell-vekt(attr(V1,"bukk",9,V2),+4.6e+1) ->;
ideell-vekt(attr(V1,"bukk",10,V2),+4.6e+1) ->;
ideell-vekt(attr(V1,"bukk",11,V2),+4.6e+1) ->;
ideell-vekt(attr(V1,"bukk",12,V2),+4.6e+1) ->;

voksen-vekt(attr(V1,"simle",V2,V3),+5.3e+1) ->;
voksen-vekt(attr(V1,"bukk",V2,V3),+4.6e+1) ->;

"-----------------------------------------------------------------------"
"                          KONDISJON"
"-----------------------------------------------------------------------"
"Henter inn en initiell vekt for tidsskritt 0;"
"Hvis brukeren starter med tidsskritt 0 gjelder default!!! Det samme er"
"tilfelle for mortalitet;"
"kondisjon(A-ttributter,N-yvekt) ->"
"Kondisjon er det samme som vekta på dyret.Her beregnes ny vekt <Nyvekt>"
"på grunnlag av en rekke miljøbetingelser.?????"

kondisjon(attr(S-ted,K-joenn,A-lder,0),V-ekt) ->
    outml("KOND")
    ideell-vekt(attr(V1,K-joenn,A-lder,V2),V-ekt)
    outml("KOND<");
kondisjon(A-ttributter,N-yVekt) ->
    outml("KOND")
    eq(attr(S-ted,K-joenn,A-lder,T-PlusEn),A-ttributter)
    val(sub(T-PlusEn,1),T)
    val(sub(A-lder,1),A-ldMinEn)
    ask-komp(kondisjon(attr(S-ted,K-joenn,A-ldMinEn,T),V-ektFoer),V-ektFoer)
    syk(A-ttributter,S)
    beite-pr-ind(A-ttributter,B-I)
    inst-forst(F)
    ideell-vekt(A-ttributter,I-deellVekt)
    vekt-rel-n-vekt(attr(V1,K-joenn,A-ldMinEn,V2),V-ektFoer,R-elVekt)
    vekst-justering(R-elVekt,V-J)
    ny-kond-f(A-ttributter,V-ektFoer,B-I,F,V-J,N-yVekt)
    ass-rein-db(asked-komp(kondisjon(A-ttributter,N-yVekt),N-yVekt))
    outml("KOND<");

"-----------------------------------------------------------------------"
```

```
"################################################"
"                KONDISJON-FUNKSJONEN"
"################################################"
"ny-kondisjon(Attributter,Vf,Bi,F,Vj,Nyvekt) ->"
"nykondisjon beregner ny kondisjon i årsstepp;"
"Ingen beite eller kondisjonsproblemer;"
"Problemer!!Line r påvirkning(proporsjonal);"

ny-kond-f(attr(V1,K-j,A-ld,V2),V-F,B-I,F,V-J,N-yVekt) ->
   val(inf(mul(+3.0,B-I),+1.0),0)
   eq(F,+0.0)
   val(sub(A-ld,1),A-ldMinEn)
   ideell-vekt(attr(V1,K-j,A-ldMinEn,V2),I-VNu)
   ideell-vekt(attr(V1,K-j,A-ld,V2),I-VNeste)
   val(sub(I-VNeste,I-VNu),D-iffIV)
   val(mul(add(V-F,D-iffIV),V-J),N-yVekt);
ny-kond-f(attr(V1,K-j,A-ld,V2),V-F,B-I,F,V-J,N-yVekt) ->
   val(sub(A-ld,1),A-ldMinEn)
   ideell-vekt(attr(V1,K-j,A-ldMinEn,V2),I-VNu)
   ideell-vekt(attr(V1,K-j,A-ld,V2),I-VNeste)
   val(sub(I-VNeste,I-VNu),D-iffIV)
   paadrag(B-I,F,D-iffIV,J-ustertPaadrag)
   val(mul(add(V-F,J-ustertPaadrag),V-J),N-yVekt);

"################################################"
"paadrag(Beite,Foru,DiffIv,Paadrag) ->"
"Beite ligger mellom 0 og 1/3. Foru mellom 0 og 1. DiffIv er differanse "
"vekt mellom ideellvekt Alder-1 til Alder."

paadrag(B-eite,F-oru,D-iffIV,P-aadrag) ->
   paavirkning-beite(B-eite,B)
   paavirkning-foru(F-oru,F)
   val(sub(D-iffIV,add(mul(B,mul(B-eite,D-iffIv)),mul(F,mul(F-oru,D-iffIv)))),
   P-aadrag);

paavirkning-beite(B-eite,B) ->
   val(div(+1.0,+3.0),E-ntredj)
   normaliser(E-ntredj,B-eite,N-Beite)
   bez("Beite","Kond",paavirk-beite-bez(N-Beite,B));



paavirkning-foru(F-oru,F) ->
   bez("Forurensing","Kond",<paavirk-foru-bez,F-oru,F>);


"_____________________________________________________________"
"vekst-justering(R-elVekt,V-ekstJust) ->"
"foreløbig line r vekstjustering. Potensiell vekt|kning maksvekt hvis"
"kondisjon er 0.6."

max-vekt-just(+2.0) ->;


"Proporsjonal minking av justeringsfaktoren ned til 1(Identitet);"
"Når vekt på dyret = ideell vekt så vokser dyret etter "
"nykondisjon beregningen;"

vekst-justering(R-elVekt,V-ekstJust) ->
   eq(R-elVekt,+6.0e-1)
   max-vekt-just(V-ekstJust);
vekst-justering(R-elVekt,V-ekstJust) ->
   max-vekt-just(M-axVJ)
   val(div(sub(M-axVJ,+1.0),+4.0e-1),D-iff)
   val(sub(M-axVJ,mul(D-iff,sub(R-elVekt,+6.0e-1))),V-ekstJust);

"_____________________________________________________________"

vekt-rel-n-vekt(attr(V1,K-j,A-lder,V2),V-Foer,R-ate) ->
   ideell-vekt(attr(V1,K-j,A-lder,V2),I-Vekt)
   val(div(V-Foer,I-Vekt),R-ate);

vekt-rel-voksen-vekt(attr(V1,K-j,V2,V3),V-ekt,R-ateVoksVekt) ->
   voksen-vekt(attr(V1,K-j,V2,V3),V-v)
   val(div(V-ekt,V-v),R-ateVoksVekt);

"Tilgjengelig beites innvirkning på kondisjonen."

tilgjb-til-k(G-radTilgjengeligBeite) ->;


"_____________________________________________________________"

"************************************************************"
"************************************************************"
"SYK"
"syk(attr(-,-,-,0),0);"
"henter inn initiell syk; I tidsskritt <0> er syk <0>; D.v.s at det ikke er"
"sykdomspåvirkning fra installasjoner;"
"syk(Attributter,S) ->"
"Attributter-attr(Sted,Kjoenn,Alder,Tid);"
"tar på grunnlag av kjønn, alder og tid, og beregner en sykdomsrate i"
```

```
"intervallet [0,1];"

syk(attr(V1,V2,V3,0),+0.0) -> outml("SYK");
syk(attr(S-ted,K-joenn,A-lder,T-PlusEn),N-S) ->
   outml("SYK")
   val(sub(T-PlusEn,1),T)
   val(sub(A-lder,1),A-ldMinEn)
   kond-til-s(G-radK)
   ask-komp(kondisjon(attr(S-ted,K-joenn,A-ldMinEn,T),K),K)
   foru-til-s(G-radForu)
   inst-foru(F-oru)
   val(add(mul(G-radK,K),mul(G-radForu,F-oru)),S)
   null-en("syk",S,N-S)
   ass-rein-db(asked-komp(syk(attr(S-ted,K-joenn,A-lder,T-PlusEn),N-S),N-S))
   outml("SYK<");

"_____________________________________________________________"
"################################################"
"            SYK-FUNKSJONEN"
"################################################"
" S is Grad-k*K + Grad-foru*F, "
"################################################"
"Foru-til-s er predikatet som forteller hvor stor påvirkning "
"forurensing har på SYK;"

foru-til-s(I-nnv) ->
   ask(forusykq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

"Kondisjonens innvirkning på sykdom;"

kond-til-s(I-nnv) ->
   ask(kondsykq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

"_____________________________________________________________"

"************************************************************"
"************************************************************"
"Vi antar det bare er et kalvingsområde for hver bestand.(k=beskaffenhet)"
"               Kalvingsområde"
"kalvingsomraade(S-ted,Areal,K) ->"
"sjekker om det finnes forstyrrelse på S-ted <S-ted>.Hvis ikke så er "
"kalvings- område urørt, ellers finnes ut hva slags og hvor kraftig "
"forstyrrelse. Beskaffenheten til kalvingsområde <K> blir justert "
"deretter i <forst->ko/2>. Beskaffenhet= besk(Areal,Kvalitet)."
Må vel også v re avhengig av forrige tidsstepps kalvo. beskaffenhet."

kalvingsomraade(attr(S-ted,V1,V2,T-PlusEn),B-eskaffenhet) ->
   outml("KALVO")
   not(forstyrr(S-ted))
   val(sub(T-PlusEn,1),T)
   ask-komp(kalvingsomraade(attr(S-ted,V1,V2,T-PlusEn),B-eskaffenhet),
   B-eskaffenhet)
   outml("KALVO<");
kalvingsomraade(attr(St-ed,V1,V2,T-PlusEn),B-eskaffenhet) ->
   outml("KALVO")
   val(sub(T-PlusEn,1),T)
   ask-komp(kalvingsomraade(attr(S-ted,V1,V2,T),B-eskaffenhet),B-eskaffenhet)
   total-forst-til-ko(S-ted,B-eskaffenhet)
   outml("KALVO<");

"_____________________________________________________________"
"Gående og helikopter spesielt!"

forst-til-ko(S-ted,besk(A-real,K-val)) ->
   ask(stedaiq,S)
   eq(S,S-ted)
   val(+0.0,K-val)
   val(+0.0,A-real)
   climb
   assert(asked(oedelagtkalvoq,"ja"),nil)
   down("rein");
forst-til-ko(S-ted,B-esk) ->
   rute-flyg(X,S-ted)
   antall-flyginger(A-nt)
   flyg-forst-kalv-f(A-nt,B-esk);
forst-til-ko(S-ted,B-esk) ->
   rute-flyg(S-ted,X)
   antall-flyginger(A-nt)
   flyg-forst-kalv-f(A-nt,B-esk);
forst-til-ko(S-ted,B-esk) ->
   sted-turgange(S-ted)
   ant-turgaaere(A-nt)
   tur-forst-kalv-f(A-nt,B-esk);

"total-forst-til-ko(B-esk) ->"
"går igjennom alle forstyrrelses typene og henter deres respektive"
"beskaffenheter <Areal,Kvalitet> isolert sett.  Disse blir multiplisert"
"sammen. Vi må kanskje i annen omgang ha en mer sofistikert utregning."

total-forst-til-ko(S-ted,B-esk) ->
```

```
   setof(K,forst-til-ko(S-ted,besk(A,K)),K-1)
   mult-list(K-1,K);

flyg-forst-kalv-f(A-ntFlyg,besk(A,K)) ->
   normaliser(100,A-ntFlyg,N-AntFlyg)
   bez(N-Antflyg,K);


tur-forst-kalv-f(A-nt,besk(A,K)) ->
   normaliser(1000,A-nt,N-Ant)
   bez(N-ant,K);


"_____"

forstyrr(S-ted) -> ask(stedaiq,S) eq(S,S-ted);
forstyrr(S-ted) -> ask(flygfraq,S) eq(S,S-ted);
forstyrr(S-ted) -> ask(flygtilq,S) eq(S,S-ted);
forstyrr(S-ted) -> ask(baatrutefraq,S) eq(S,S-ted);
forstyrr(S-ted) -> ask(baatrutetilq,S) eq(S,S-ted);
forstyrr(S-ted) -> ask(stedturq,S) eq(S,S-ted);

"forstyrr(S-ted) -> ask(q,S),eq(S,S-ted)."
"_____"

"**********************************************************"
"**********************************************************"
"Reproduksjon"
"reproduksjon(Attributter,R) ->"
"Attributter=attr(Sted,Kjoenn,Alder,Tid);"
"Gir reproduksjonsrate <R> for simlene med alder <Tid> (T);"

reproduksjon(A-ttributter,N-R) ->
   outml("REPRO")
   syk-til-rp(G-radS)
   syk(A-ttributter,S)
   kond-til-rp(G-radK)
   kondisjon(A-ttributter,V-ekt)
   kondis-repro(A-ttributter,V-ekt,R-eproRate)
   antall-km-flyg(K-m)
   ant-turgaaere(A-ntTurer)
   repro-f(G-radS,S,G-radK,R-eproRate,K-m,A-ntTurer,R)
   nulle-en("repro",R,N-R)
   ass-rein-db(asked-komp(reproduksjon(A-ttributter,N-R),N-R))
   outml("REPRO<");

"_____"

repro-f(G-radS,S,G-radK,R-eproRate,K-m,A-ntall,R) ->
   val(add(mul(G-radS,S),mul(G-radK,R-eproRate)),R);

"_____"
"kondis-repro(attr(V1,Kj,V2,V3),V-ekt,R-epro) ->"
"Kondisjonens innvirkning på reproduksjonen;"
"Line r reproduksjon stigning fra 70% av voksenvekt til Voksenvekt;"
"0.5 i repro til 0.95;"

kondis-repro(attr(V1,K-j,V2,V3),V-ekt,R-epro) ->
   vekt-rel-voksen-vekt(attr(V1,Kj,V2,V3),V-ekt,P)
   val(inf(P,+7.0e-1),1)
   val(+0.0,R-epro);
kondis-repro(attr(V1,Kj,V2,V3),V-ekt,N-Repro) ->
   vekt-rel-voksen-vekt(attr(V1,Kj,V2.V3),V-ekt,P)
   val(inf(P,101),1)
   val(add(+5.0e-1,mul(sub(P,+7.0e-1),div(+3.0,+2.0))),R-epro)
   null-en("kondis-repro",R-epro,N-Repro);

"Repro is (P-70)*(3/200) + 0.5;"
"ref Øritslands notater;"

kond-til-rp(I-nnv) ->
   ask(kondreproq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

"_____"
"Sykdoms innvirkning på Reproduksjonen"

syk-til-rp(I-nnv) ->
   ask(sykreproq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

"_____"

"**********************************************************"
"**********************************************************"
"           VANDRINGER"
"vandringer(Attributter,V) ->"
"gir rate <V> for hvor mange dyr som vandrer i tid <T> på grunnlag "
"av forstyrrelse og kondisjon i tid <T>;"

vandringer(A-ttributter,V) ->
   outml("VANDR")
```

```
   forst-til-v(G-radForst)
   kond-til-v(G-radK)
   inst-forst(F)
   ask-komp(kondisjon(A-ttributter,V-ekt),V-ekt)
   vekt-rel-n-vekt(A-ttributter,V-ekt,R-elVekt)
   vandr-f(G-radForst,G-radK,F,R-elVekt,V)
   ass-rein-db(asked-komp(vandringer(A-ttributter,V),V))
   outml("VANDR<");

"################################################"
"            VANDRING-FUNKSJONEN"
"################################################"

vandr-f(G-rF,G-rK,F,R-V,V) ->
   val(add(mul(G-rF,F),mul(G-rK,R-V)),V);

"################################################"
"Februar, mars, april; Bare voksne dyr! (70% av asymptote vekt);"
"Kondisjonens innvirkning på vandringer;"

kond-til-v(I-nnv) ->
   ask(kondvandrq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

forst-til-v(I-nnv) ->
   ask(forstvandrq,S-Innv)
   str-pros-real(S-Innv,I-nnv);

"_____________________________________________________________"

"************************************************************"
"************************************************************"
"tilgj-beite(T+1,B) ->"
"Henter inn areal reinvegetasjon i området, reinens arealbehov og"
"beregner nytt beite <B>;"

tilgj-beite(attr(S-ted,K-j,A-ld,T-PlusEn),<B-eite,B-ehov>) ->
   outml("TILGJBEITE")
   val(sub(T-PlusEn,1),T)
   val(sub(A-ld,1),A-ldMinEn)
   areal-rein-veg(attr(S-ted,V1,V2,T-PlusEn),A-real)
   beite-behov(attr(S-ted,K-j,A-ldMinEn,T),B-ehov)
   tilgj-b(A-real,B-ehov,B-eite)
   ass-rein-db(asked-komp(tilgj-beite(attr(S-ted,V1,V2,T-PlusEn),<B-eite,B-ehov>
   ),<B-eite,B-ehov>))
   outml("TILGJBEITE<");

"tilgj-b(Areal,Behov,Beite) ->"
"beregner baitegrunnlaget <Beite> i areal for reinen;"

tilgj-b(A-real,B-ehov,B-eite) ->
   val(sub(B-ehov,A-real),D-iff)
   val(inf(+0.0,D-iff),1)
   val(sub(A-real,D-iff),B-eite);
tilgj-b(A-real,B-ehov,A-real) -> val(inf(+0.0,sub(A-real,B-ehov)),1);

"Vi antar ingen minking av beite når beitetrykket er mindre enn"
"eller lik kapasiteten; Klima er begrensende faktor for vegetasjonen;"
"beregner baitegrunnlaget i areal for reinen;  "

areal-rein-veg(attr(S-ted,V1,V2,T-PlusEn),A-real) ->
   ask-komp(areal-rein-veg(attr(S-ted,V1,V2,T-PlusEn),A-real),A-real);

"_____________________________________________________________"

;End world: rein


;

;
```

```
"World:tools"
"*********************************************************"
"Vektøy og brukergrensesnitt for DAKON"
"*********************************************************"
"Matcher flere tekster til hver kommando;  For eksempel kan brukeren "
"skrive <avslutt>  <av>  <a>  <sl>  <exit>  <quit> eller <q> for å avslutte "
"programnivå; Det er her lett å tilpasse andre språk!"

" Avslutter programnivå; "

avslutt("a") ->;
avslutt("avslutt") ->;
avslutt("av") ->;
avslutt("slutt") ->;
avslutt("sl") ->;
avslutt("exit") ->;
avslutt("quit") ->;
avslutt("q") ->;

endre-faktum("ef") ->;
endre-faktum("endrefaktum") ->;
endre-faktum("endre") ->;
endre-faktum("change") ->;
endre-faktum("ch") ->;
endre-faktum("endr") ->;

feltarbeid("feltarb") ->;
feltarbeid("feltarbeid") ->;
feltarbeid("felt") ->;
feltarbeid("fa") ->;
feltarbeid("feltarbeid") ->;

"fjerne alle fakta med et hode "
"abolish(Hode) -> ;   "

fjern-alle-fakta("faf") ->;
fjern-alle-fakta("fjern-alle") ->;
fjern-alle-fakta("fjerna") ->;
fjern-alle-fakta("abolish") ->;
fjern-alle-fakta("removeall") ->;
fjern-alle-fakta("rma") ->;
fjern-alle-fakta("deleteall") ->;
fjern-alle-fakta("dala") ->;

" ——''——    fjerne    ——''——"

fjern-database("fdb") ->;

fjern-faktum("ff") ->;
fjern-faktum("fjern") ->;
fjern-faktum("fjernfaktum") ->;
fjern-faktum("retract") ->;
fjern-faktum("suppress") ->;
fjern-faktum("remove") ->;
fjern-faktum("rm") ->;
fjern-faktum("delete") ->;
fjern-faktum("del") ->;

" laster inn tidligere fakta "

last-inn-state("load") ->;
last-inn-state("lastinn") ->;
last-inn-state("li") ->;
last-inn-state("ld") ->;
last-inn-state("last") ->;

" ——''——    legge til  ——''——"

legg-til-faktum("leggtil") ->;
legg-til-faktum("legg-til-faktum") ->;
legg-til-faktum("lt") ->;
legg-til-faktum("ltf") ->;
legg-til-faktum("assert") ->;
legg-til-faktum("ass") ->;

"Negative svar -> ;  "

negativ("n") ->;
negativ("nei") ->;
negativ("ikke") ->;
negativ("feil") ->;
negativ("aldri") ->;
negativ("umulig") ->;
negativ("uenig") ->;

" Positive svar -> ;  "

positiv("ja") ->;
positiv("j") ->;
positiv("riktig") ->;
positiv("stemmer") ->;
positiv("korrekt") ->;
```

```
positiv("enig") ->;
positiv("ok") ->;

"Starter selve analysesystemet for rein;"

run-analysis("start") ->;
run-analysis("dakon") ->;
run-analysis("run") ->;
run-analysis("go") ->;
run-analysis("x") ->;
run-analysis("star") ->;

"Starter innlesing av installasjons fakta;"

start-ai("ai") ->;
start-ai("s") ->;
start-ai("sti") ->;
start-ai("startai") ->;
start-ai("installasjon") ->;
start-ai("stinst") ->;

"Går til vedlikeholds-programnivå;"
"På dette nivået kan programmet forandres"

vedlikehold("vedlikehold") ->;
vedlikehold("v") ->;
vedlikehold("ved") ->;

vis-fakta("visfakta") ->;
vis-fakta("list") ->;
vis-fakta("vis") ->;
vis-fakta("v") ->;
vis-fakta("vf") ->;
vis-fakta("ls") ->;

"-------------------------------------------------------------"
"Lager konstant for spørsmål og svar (modularisering); Variablene som "
"f. eks;  <andelsniksk yttq> ekspanderes til tekster (Hvor stor andel"
" personal er sniksk yttere>) som skrives på skjermen; "
"(De er organisert alfabetisk);  Det er lett å gå inn å forandre på "
"tekstene her;  Dette er både nyttig under programmering og"
" oppgradering  vedlikehold;  Lett å forandre språk; "
"-------------------------------------------------------------"

questioncode(alderreinq,"Alder   rein (simle:0; 16/bukk:0; 12)>") ->;
questioncode(andelsniksk yttq,"Hvor stor andel personal er sniksk yttere>") ->;
questioncode(andrexderivq,"Derivert X-verdi   slutt>") ->;
questioncode(andreyderivq,"Derivert Y-verdi   slutt>") ->;
questioncode(andrexkordq,"X-koordinat   slutt>") ->;
questioncode(andreykordq,"Y-koordinat   slutt>") ->;
questioncode(antbaatturerq,"Antall båttturer>") ->;
questioncode(antdetonq,"Antall detoneringer>") ->;
questioncode(antflygq,"Hvor mange flyginger pr.  døgn>") ->;
questioncode(antpersq,"Hvor mange personer er koplet til virksomheten>") ->;
questioncode(antturq,"Turgåere pr døgn>") ->;
questioncode(arealreinvegq,"Areal reinvegetasjon(Kvm)>") ->;
questioncode(baatrutefraq,"Båtrute fra>") ->;
questioncode(baatrutetilq,"Båtrute til>") ->;
questioncode(dwtq,"DWT>") ->;
questioncode(fjernalle-faktaq,"Fjern alle fakta (<hode(X  Y  Z);  >)>") ->;
questioncode(fjernfaktumq,"Fjern fakta (<hode(X  Y  Z);  >)>") ->;
questioncode(fjerndenneq,"Vil du fjerne denne(ja/nei)>") ->;
questioncode(flygfraq,"Helikopterrute fra>") ->;
questioncode(flygtilq,"Helikopter rute til") ->;
questioncode(foerstexderivq,"Derivert X-verdi   start>") ->;
questioncode(foersteyderivq,"Derivert Y-verdi   start>") ->;
questioncode(foerstexkordq,"X-koordinat   start>") ->;
questioncode(foersteykordq,"Y-koordinat   start>") ->;
questioncode(forstvandrq,"Forstyrrelsers innvirkning på vandring(0  1)>") ->;
questioncode(fortsettq,"Fortsett(ja/nei)>") ->;
questioncode(forusykq,"Forurensings innvirkning på syk(0  1)>") ->;
questioncode(hvilkenverdenq,"Legge til regel i hvilken verden>") ->;
questioncode(kbkavfq,"Kbkm dumpet avfall>") ->;
questioncode(kbkborslamq,"Kbkm boreslam>") ->;
questioncode(kbkmavfallq,"Kbkm forbrent avfall>") ->;
questioncode(kgdynamittq,"Kg;  dynamitt>") ->;
questioncode(kjoennreinq,"Kjønn   rein(simle/bukk>") ->;
questioncode(kmflygq,"Antall km;  flyging>") ->;
questioncode(kmprodq,"Km;  produksjon>") ->;
questioncode(kondmortq,"Kondisjonens innvirkning på mortaliteten(0  1)>") ->;
questioncode(kondq,"kondisjon(Kjoenn  Alder  Tid  Vekt); >") ->;
questioncode(kondreproq,"Kondisjonens innvirkning på reproduksjonen(0  1)>") ->;
questioncode(kondsykq,"Kondisjonens innvirkning på sykdom(0  1)>") ->;
questioncode(kondvandrq,"Kondisjonens innvirkning på vandring(0  1)>") ->;
questioncode(kalvoq,"kalvingsområde(Sted  Areal  Tid  Beskaffenhet); >") ->;
questioncode(leggtilhodeq,"Regelhode(<hode(X  Y);  >)>") ->;
questioncode(leggtilhaleq,"Regelhale(<halel(X)  hale2(Y  Z);  >(evnt <true; >)>"
   ) ->;
questioncode(ltrbensinq,"Liter bensin>") ->;
questioncode(ltrborvskq,"Liter spilt borev ske") ->;
questioncode(ltrdieselq,"Liter diesel>") ->;
questioncode(ltroljq,"Liter forbruk olje>") ->;
```

```
questioncode(ltrsmoq,"Liter spilt smøreolje>") ->;
questioncode(ltrspolq,"Liter spilt olje>") ->;
questioncode(lovligbeskattq,"Tillatt beskattningsrate>") ->;
questioncode(mortq,"mortalitet(Kjønn  Alder  Tid  Mort); >") ->;
questioncode(navnaiq,"Gi navn på installasjon>") ->;
questioncode(navnkompq,"Navn på komponent(rein/kondisjon/etc)>") ->;
questioncode(oedelagtkalvoq,"Er kalvingsområde ødelagt>") ->;
questioncode(predq,"predasjon(Kjønn  Alder  Tid  Predasjon); >") ->;
questioncode(prevtilpredq,"polarrev-til-pr(Alder  Grad-p); >") ->;
questioncode(reinq,"rein(simle/bukk  Alder  Tid  Antall-rein); >") ->;
questioncode(reproq,"reproduksjon(Alder  Tid  Repro-rate); >") ->;
questioncode(sluttidaarq,"Sluttid (89)>") ->;
questioncode(sluttidmndq,"Sluttid(jan)>") ->;
questioncode(starttidaarq,"Starttid (89)>") ->;
questioncode(starttidmndq,"Starttid (jan)>") ->;
questioncode(stedaiq,"Sted  installasjon>") ->;
questioncode(stedreinq,"Sted  rein>") ->;
questioncode(stedturq,"Hvor ferdes folk>") ->;
questioncode(svarlovligtypeq,"Svar med lovlig type>") ->;
questioncode(svarjaneiq,"Svar ja eller nei>") ->;
questioncode(sykmortq,"Syks innvirkning på mortaliteten(0  1)>") ->;
questioncode(sykq,"syk(Kjønn  Alder  Tid  Syk); >") ->;
questioncode(sykreproq,"Syks innvirkning på reproduksjonen(0  1)>") ->;
questioncode(tidsstepq,"Tidsstepp>") ->;
questioncode(tilgjbeiteq,"tilgj-beite(Tid  Beiteareal); >") ->;
questioncode(typeaiq,"Hva slags virksomhet>") ->;
questioncode(typebaatq,"Båttype>") ->;
questioncode(typeboreutstyrq,"Hva slags boreutstyr skal brukes>") ->;
questioncode(typeterrengq,"Hva slags terrengtype ligger inst. i>") ->;
questioncode(typevedlikeholdq,"Hva slags vedlikehold>") ->;
questioncode(vandrq,"vandringer(Kjønn  Alder  Tid  Vandringsrate); >") ->;
questioncode(visfaktumq,"Gi faktum som skal vises(<hode(X  Y);  >)>") ->;

"explain(X) kan brukes for å forklare hvis bruker ikke taster godkjent svar"
"_____"
"                              TYPESJEKKING"
"Her angis explisitt hva som er lovlige svar på spørsmål;  "
"For eksempel er alle 12 måneder angitt;  Alfabetisk etter type; "
"_____"

lovlig-type(boreutstyr,"svada") ->;
lovlig-type(kjoenn,"simle") ->;
lovlig-type(kjoenn,"bukk") ->;
lovlig-type(sted,"edgeoya") ->;
lovlig-type(tidmnd,"jan") ->;
lovlig-type(tidmnd,"feb") ->;
lovlig-type(tidmnd,"mars") ->;
lovlig-type(tidmnd,"apr") ->;
lovlig-type(tidmnd,"mai") ->;
lovlig-type(tidmnd,"jun") ->;
lovlig-type(tidmnd,"jul") ->;
lovlig-type(tidmnd,"aug") ->;
lovlig-type(tidmnd,"sep") ->;
lovlig-type(tidmnd,"okt") ->;
lovlig-type(tidmnd,"nov") ->;
lovlig-type(tidmnd,"des") ->;
lovlig-type(tidaar,A) ->
   string-integer(A,I)
   val(inf(I,0),0)
   val(inf(I,100),1);
lovlig-type(terreng,"strand") ->;
lovlig-type(terreng,"dal") ->;
lovlig-type(terreng,"bre") ->;
lovlig-type(terreng,"fjell") ->;
lovlig-type(virksomhet,"landseismikk") ->;
lovlig-type(virksomhet,"marinseismikk") ->;
lovlig-type(virksomhet,"oljeleting") ->;
lovlig-type(virksomhet,"oljeboring") ->;
lovlig-type(virksomhet,"feltarbeid") ->;

"################################################"
"Kommando-løkka. Programmet startes ved "
">?- dakon>; "
"################################################"
"dakon -> "
"Skriver ut et prompt   leser så inn en kommando og utfører denne;"

dakon ->
   outm("**********DAKON ---> Data-assistert konsekvensanalyse**********")
   line
   line
   dakonhjelp
   line
   dakonprompt
   in-word(J-ob,X)
   line
   dakonjob(J-ob);

"dakonjob(J-ob) ->"
"Avhengig av hva <Job> er instansiert til avsluttes programmet "
"analysen starter eller programmet går til neste nivå for vedlikehold av"
"programmet; "
```

```
dakonjob(J-ob) -> last-inn-state(J-ob) dakon /;
dakonjob(J-ob) -> avslutt(J-ob) /;
dakonjob(J-ob) -> start-ai(J-ob) ai dakon /;
dakonjob(J-ob) -> vedlikehold(J-ob) vedlikeh dakon /;
dakonjob(J-ob) -> run-analysis(J-ob) run-analysis-rein dakon /;
dakonjob("h") -> dakonhjelp dakon /;
dakonjob(I-kkedef) -> outm("Finner ikke kommando!!") line dakon /;

dakonprompt -> outm("dakon|>");

dakonhjelp ->
   outm("Svar med:")
   line
   line
   outml("a       : Avslutt")
   outml("li      : Last Inn forrige memory-status")
   outml("star    : STart Analyse Rein")
   outml("sti     : STart innlesing om Installasjon")
   outml("v       : Vedlikehold av program/database")
   outml("h       : Hjelp")
   line;

"_____________________________________________________________________"
"vedlikehold ->"
"Skriver ut et prompt, leser inn en kommando og utfører denne; "

vedlikeh ->
   vedlikeholdhjelp
   line
   vedlikeholdprompt
   in-word(V-edlikeholdJob,Y)
   line
   vedlikeholdjob(V-edlikeholdJob);

"vedlikeholdjob(V-edlikeholdJob) ->"
"Avhengig av hva <Job> er instansiert til avsluttes vedlikehold"
"(tilbake til hovednivå)   fakta blir lagt til databasen osv; "

vedlikeholdjob(J-ob) -> avslutt(J-ob) /;
vedlikeholdjob(J-ob) -> legg-til-faktum(J-ob) legg-til vedlikeh /;
vedlikeholdjob(J-ob) -> fjern-database(J-ob) fjern-db vedlikeh /;
vedlikeholdjob(J-ob) -> fjern-faktum(J-ob) fjern vedlikeh /;
vedlikeholdjob(J-ob) -> fjern-alle-fakta(J-ob) fjern-alle vedlikeh /;
vedlikeholdjob(J-ob) -> vis-fakta(J-ob) vis vedlikeh /;
vedlikeholdjob(J-ob) -> endre-faktum(J-ob) endre vedlikeh /;
vedlikeholdjob(J-ob) -> vedlikeholdhjelp vedlikeh /;
vedlikeholdjob(I-kkedef) -> outm("Finner ikke kommando!!") vedlikeh /;

vedlikeholdprompt -> outm("vedlh|>");

vedlikeholdhjelp ->
   outm("Svar med:")
   line
   line
   outml("a       : Avslutt vedlikehold")
   outml("ltf     : Legg Til Faktum")
   outml("ff      : Fjern Faktum")
   outml("fdb     : Fjern DataBase")
   outml("faf     : Fjern Alle Fakta")
   outml("vf      : Vis Fakta")
   outml("ef      : Endre Faktum")
   outml("h       : Hjelp")
   line;

"_____________________________________________________________________"
"Utfører modifikasjonen av databasen"

legg-til ->
   wrt-rd(hvilkenverdenq,V-erden)
   down(V-erden)
   outm("Skriv inn regler i world ")
   out(V-erden)
   outml(" og slutt med en ekstra ';'")
   insert
   climb("tools");

fjern ->
   wrt-rd-term(fjernfaktumq,H-ode)
   not(ikke-fjern-en(H-ode));

fjern-alle ->
   wrt-rd-term(fjernallefaktaq,H-ode)
   rule(H-ode,T)
   suppress(1);

fjern-db ->
   rule-nb(asked,N)
   find-rule(asked)
   suppress(N)
   down("rein")
   kill-subworld("rein-db")
```

```
    new-subworld("rein-db",3000)
    climb outml("Database fjernet!");

vis -> wrt-rd-term(visfaktumq,H-ode) not(ikke-vis-frem(H-ode));

"For å presse fram clause til å hente alle Hoder som matcher"
"(forskjellige instansieringer); "

endre -> vis fjern legg-til;

ikke-vis-frem(H-ode) ->
    rule(H-ode,T)
    out(H-ode)
    outm(" -> ")
    out(T)
    outm("; ")
    line
    fail;

"ikke-fjern-en(H-ode) ->"
"feiler for å få med seg alle instansieringer av hode; "
"Brukeren velger hvilke som skal fjernes"

ikke-fjern-en(H-ode) ->
    rule(H-ode,T)
    outm(H-ode)
    outm(" -> ")
    out(T)
    outm("; ")
    line
    wrt-if(fjerndenneq)
    find-rule(H-ode)
    suppress(1)
    fail;

"_____"
"################################################"
"SLUTT KOMMANDOLØKKA"
"################################################"
"_____"
"       Brukergrensenitt   hjelpeprosedyrer"
"_____"

ass-rein-db(T) ->
    down("rein")
    down("rein-db")
    assert(T,nil)
    climb
    climb;

inst-foru(F-oru) ->
    forurensing(F-oru);

inst-forst(F) ->
    forstyrrelse(F);

inst-ferd(F) ->
    ferdsel(F);

sted(S-ted) -> asked(stedaiq,S-ted);
sted(S-ted) ->
    not(asked(stedaiq,S-ted))
    down("sted")
    do-sted(S-ted)
    climb
    assert(asked(stedaiq,S-ted),nil);

"NOT"

not(P) -> P / fail;
not(P) ->;

number(N) -> real(N);
number(N) -> integer(N);

str-pros-real(S-trPros,R-eal) ->
    string-integer(S-trPros,P-rosInt)
    val(div(float(P-rosInt),+1.0e+2),R-eal);

plus-list(nil,0) ->;
plus-list(X.L,Y) -> plus-list(L,Z) val(add(X,Z),Y);

mult-list(nil,1) ->;
mult-list(X.L,Y) -> mult-list(L,Z) val(mul(X,Z),Y);

"Sjekker om X ligger mellom 0 og 1. Hvis mindre enn 0 gi svar 0."
"Hvis større enn 1 gi svaret 1. Ellers gi X som svar"

null-en(N-avn,X,N-ormalX) ->
    val(inf(+1.0,X),1)
    outm(N-avn)
    outm("=")
```

```
   out(X)
   outml(" out of range (0,1)!")
   eq(N-ormalX,+1.0);
null-en(N-avn,X,N-ormalX) ->
   val(inf(X,+0.0),1)
   outm(N-avn)
   outm("=")
   out(X)
   outml(" out of range (0,1)!")
   eq(N-ormalX,+0.0);
null-en(N-avn,X,X) -> out(N-avn) outm("=") out(X);

"_____________________________________________________________"
"INPUT - OUTPUT"
"_____________________________________________________________"
"Innhenting av fakta (caching);"
"Spør spørsmål Q, leser svar A og legger til databasen hvis denne regel "
"eller dette faktum ikke er i databasen. Ellers returneres regel eller "
"faktum gjennom A; "
"Regel eller faktum blir lagt ned i databasen på følgende form:"
"asked(<regel>) eller asked(faktum); "
"_____________________________________________________________"

ask(Q-Code,A) -> asked(Q-Code,A);
ask(Q-Code,A) ->
   not(asked(Q-Code,A))
   questioncode(Q-Code,Q)
   outm(Q)
   outm("?")
   in-word(A2,Y)
   line
   ask2(Q,Q-Code,A2,A);

"1.Bruker trenger forklaring"
"2.For bruk når vi vil ha et spesielt svar, nemlig A"
"2.Faktum blir lagt ned i databasen; "
"3.Ber om } forandre regler i databasen; "

ask2(Q,Q-Code,"?",A) -> explain(Q-Code) ask(Q-Ccode,A);
ask2(Q,Q-Code,A1,A1) -> assert(asked(Q-Code,A1),nil);
ask2(Q,Q-Code,V,A) -> vedlikehold(V) vedlikeh ask(Q-Code,A);

ask3(Q,Q-Code,A,A) -> assert(asked(Q-Code,A),nil);

"_____________________________________________________________"
"               SPØR MED TYPESJEKK"
"_____________________________________________________________"
"type-ask(Q,  T,  B) ->"
"Spør om en type (Q) og sjekker at det svares (A) med en type (T) som"
" er lovlig; outm(X)  outm(Y) kan ikke brukes da den inneholder "
"første svaret som ikke trenger v re riktig; <asked(typeq  type  X)>"
"i databasen inneholder riktig svar; "

type-ask(Q,T,A) -> ask(Q,A) svar-lovlig-type(Q,T,A);

"svar-lovlig-type(Q,  T,  A) ->"
"(A) er lovlig type (T) hvis det matcher med lovlig-type/2 tabell; "
"Hvis ikke fjernes dette fra databasen og spørsmålet stilles en gang til; "

svar-lovlig-type(Q,T,A) -> lovlig-type(T,A) /;
svar-lovlig-type(Q-Kode,T-ype,S-var) ->
   wrt-rd(svarlovligtypeq,A)
   rule(asked(Q-Kode,S-var),nil)
   suppress(1)
   assert(asked(Q-Kode,A),nil)
   svar-lovlig-type(Q-Kode,T-ype,A);

"_____________________________________________________________"
"         SPØR JA/NEI SPØRSMÅL"
"_____________________________________________________________"
"ask-if(Q) ->"
"Spør ja/nei spørsmål;  Hvis svaret er positivt (positive(A)) nås målet;"
"Hvis svaret er negativt feiles (fail);  Er svaret ingen av delene prøves p)"
"nytt!"

ask-if(Q) -> ask(Q,A) positive-answer(Q,A);

"Hvis positivt svar, legg til databasen;  Hvis hverken positivt eller"
"negativt svar, spør om igjen; "

positive-answer(Q,A) -> positiv(A);
positive-answer(Q-Code,A) ->
   not(negativ(A))
   not(positiv(A))
   wrt-rd(svarjaneiq,A2)
   rule(asked(Q-Code,A),nil)
   suppress(1)
   assert(asked(Q-Code,A2),nil)
   positive-answer(Q-code,A2);

"_____________________________________________________________"
"            SPØR ETTER KOMPONENT"
```

```
"-------------------------------------------------------------------"
"ask-komp(K-omp) ->"
"sender med en halvinstansiert komponent som spørsmål"
"(f;  eks;  <kondisjon(simle,  12 , 2 , Vekt)>"
"og intansierer fullstendig i <Komp>  hvis den er i databasen; "
"Ellers spørres bruker;  Q er på formen rein(Attributter  X)   hvor det"
"er X'en vi skal finne; Vi trenger ikke gå ned i REIN på den andre"
"da det allerede er gjort på det første forsøket! Det er dessverre"
"ingen backtrackingseffekt på DOWN og CLIMB."

ask-komp(Q,K-omp) ->
   down("rein")
   down("rein-db")
   asked-komp(Q,K-omp)
   climb
   climb;
ask-komp(Q,K-omp) ->
   not(asked-komp(Q,K-omp))
   climb
   climb
   split(Q,L-istQ)
   wrt-komp(L-istQ)
   arg(1,L-ist-Q,H-L)
   outm("::gi ")
   out(H-L)
   outm(">")
   in-word(S-Maaltall,P-rosentMaltall)
   val(div(float(P-rosentMaltall),+1.0e+2),M-rate)
   line
   ask-komp2(Q,M-rate,K-omp);

"ask-komp2(Q , T-all,  K-omp) ->"
"sjekker om <Tall> er et tall;  I så fall blir tallet sendt tilbake gjennom"
"<Komp> opp  og ut av <Komp> i ask-komp; "
"Faktum blir lagt ned i databasen; "
"Ber om å forandre regler i databasen; "

ask-komp2(Q,"?",K-omp) ->
   explain(Q)
   ask-komp(Q,K-omp);
ask-komp2(Q,V,K-omp) ->
   vedlikehold(V)
   vedlikeh
   ask-komp(Q,K-omp);
ask-komp2(Q,T-all,K-omp1) ->
   number(T-all)
   val(T-all,K-omp1)
   down("rein")
   down("rein-db")
   assert(asked-komp(Q,K-omp1),nil)
   climb
   climb;

"ask-attr(N-avn,  attr(S-t , K-j,  A-ld , T)) -> "
"henter attributtene til komponent <Navn> som må v re på formen"
"rein  kondisjon etc; "

wrt-komp(N-avn.L-iste) ->
   outm("Komponent: ")
   wrt(N-avn)
   line
   arg(1,L-iste,A-ttr)
   split(A-ttr,A-ttrListe)
   arg(2,A-ttrListe,S-ted)
   outm("Sted      : ")
   wrt-str(S-ted)
   line
   arg(3,A-ttrListe,K-j)
   outm("Kjønn     : ")
   wrt-str(K-j)
   line
   arg(4,A-ttrListe,A-ld)
   outm("Alder     : ")
   wrt(A-ld)
   line
   arg(5,A-ttrListe,T)
   outm("Tidsstepp: ")
   wrt(T)
   line;

"-------------------------------------------------------------------"
"SKRIVUT OG LES (SKJERM)"
"uten modifikasjon av database; "
"-------------------------------------------------------------------"

wrt-str(S) ->
   bound(S)
   outm(S);
wrt-str(S) ->
   free(S)
   outm("X");
```

```
wrt(T) ->
   bound(T)
   out(T);
wrt(T) ->
   free(T)
   outm("X");

wrt-rd-term(Q-Kode,A) ->
   questioncode(Q-Kode,Q)
   outm(Q)
   outm("?")
   in(A)
   in-char(C)
   line;

wrt-rd(Q-Code,A) ->
   questioncode(Q-Code,Q)
   outm(Q)
   outm("?")
   in-word(A,Y)
   line;

wrt-if(Q) -> wrt-rd(Q,A) pos-answ(Q,A);

pos-answ(Q,A) -> positiv(A);
pos-answ(Q-Code,A) ->
   not(negativ(A))
   not(positiv(A))
   wrt-rd(svarjaneiq,A2)
   pos-answ(Q-Code,A2);

"-------------------------------------------------------------------"
"              SPØR MED TYPESJEKK"
"-------------------------------------------------------------------"
"type-ask(Q , T,  B) ->"
"Spør om en type (Q) og sjekker at det svares (A) med en type (T) som er"
"lovlig; A kan ikke brukes da den inneholder første svaret som ikke "
"trenger v re riktig; "

type-wrt-rd(Q,T,A) -> wrt-rd(Q,A) read-lovlig-type(Q,T,A);

"read-lovlig-type(Q,  T,  A) ->"
"(A) er lovlig type (T) hvis det matcher med lovlig-type/2 tabell; "
"Hvis ikke stilles spørsmålet  en gang til; "

read-lovlig-type(Q,T,A) -> lovlig-type(T,A) /;
read-lovlig-type(Q-Kode,T-ype,S-var) ->
   wrt-rd(svarlovligtypeq,A)
   read-lovlig-type(Q-Kode,T-ype,A);

"-------------------------------------------------------------------"
"Uavhengig kontra avhengig fakta "
"Uavhengig fakta med mulighet som akkumuleres."
"Flere betingelser og krav må oppfylles for at venstresida skal gjelde; "
"indep-andcombine([P]  P); "

"################################################"
"                FUNKSJONER"
"################################################"
"normaliser(M-ax , X , N-ormalisertX) ->"
"normaliserer <X> til å v re i intervallet [0  1];  Når X->Max så er"
"NormalisertX = 1; "
"Hvis funksjonen ligger klar benyttes denne, ellers lages en ny."
"Første klausul passer på å gå ned i bezier"

bez(X-Akse,Y-Akse,<N-avn,X',Y>) ->
   down("rein")
   down("rein-db")
   tabell(<N-avn,P1,P2,P3,P4>)
   climb
   climb
   find-y-bez(P1,P2,P3,P4,+0.0,X',F-aktiskX,Y);
bez(X-Akse,Y-Akse,<N-avn,X',Y>) ->
   climb
   climb
   down("bezier")
   do-bezier(X-Akse,Y-Akse,P1,P2,P3,P4)
   climb
   down("rein")
   down("rein-db")
   assert(tabell(<N-avn,P1,P2,P3,P4>),nil)
   climb
   climb
   find-y-bez(P1,P2,P3,P4,+0.0,X',F-aktiskX,Y);

normaliser(M-ax,X,N-ormalisertX) ->
   val(abs(X),A-bsX)
   val(inf(M-ax,A-bsX),0)
   val(div(A-bsX,M-ax),N-ormalisertX);
normaliser(M-ax,X,N-ormalisertX) ->
   val(abs(X),A-bsX)
   val(inf(M-ax,A-bsX),1)
```

```
   val(+1.0,N-ormalisertX);

"DATABASE"

;End world: tools


;
```

```
to-begin ->
   new-subworld("tools",15000)
   insert("Svalbard:dakonprogram:tools.to")
   new-subworld("installasjon",10000)
   insert("Svalbard:dakonprogram:installasjon.to")
   climb("tools")
   new-subworld("rein",15000)
   insert("Svalbard:dakonprogram:rein.to")
   new-subworld("rein-db",3000)
   climb("rein")
   climb("tools")
   new-subworld("bezier",4000)
   insert("Svalbard:dakonprogram:bezdraw.to")
   climb("tools")
   new-subworld("sted",3500)
   insert("Svalbard:dakonprogram:svalbardsted.to")
   climb("tools")
   dakon;


;End world: Normal

"Starter opp programmet direkte"
"laster inn Svalbard kart inn i grafisk vindu"

do-sted(S-ted) ->
   find-sted
   rule(zone(I),nil)
   suppress(1)
   assert(zone(0),nil)
   message(I,S-ted)
   /;

find-sted ->
   set-window("graphic",1,30,50,420,300)
   set-window("console",0)
   output("graphic")
   gr-load("Svalbard:macpaints:svalbardpaintfill")
   gr-draw-buttons(button-set1)
   gr-moveto(240,150)
   gr-text(3,18,3.4.6.nil)
   outml("Svalbard")
   gr-text(4,12,nil)
   block(end,always(steds-angivelse));
find-sted ->
   set-window("graphic",0)
   set-window("console",1)
   output("console");

click-test ->
   gr-click(1,X,Y)
   gr-button-hit(button-set1,<X,Y>,A)
   action(A);

action(edit) -> edit;
action(stop) ->
   set-window("graphic",0)
   set-window("console",1)
   output("console")
   block-exit(16);
action(retry) ->;
action(ok) ->
   set-window("graphic",0)
   set-window("console",1)
   output("console")
   block-exit(end);

in-part(P,I) ->
   rect(I,R)
   gr-in-rect(P,R)
   /;
in-part(P,0) ->;

steds-angivelse ->
   gr-getmouse(X,Y,B)
   show-assert(X,Y,B);

show-assert(X,Y,B) ->
   eq(B,0)
   in-part(<X,Y>,N)
   show(N);
show-assert(X,Y,B) ->
   eq(B,1)
   in-part(<X,Y>,N)
   show(N)
   click-test;

show(I) ->
   zone(I)
   /;
show(I) ->
```

```
    zone(J)
    restaure(J)
    rect(I,X)
    InvertRect(X)
    assert(zone(I),nil)
    put-message(I)
    /;
show(0) ->
    assert(zone(0),nil)
    put-message(0);

InvertRect(R) -> gr-rect(3,R);

EraseRect(R) -> gr-rect(2,R);

put-message(N) ->
    EraseRect(<0,190>.<420,300>)
    message(N,M)
    gr-moveto(40,200)
    outml(M);

restaure(I) ->
    rule(zone(I),nil)
    suppress(1)
    rect(I,X)
    InvertRect(X)
    /;
restaure(I) ->;

always(P) ->
    repeat
    P
    fail;

repeat ->;
repeat -> repeat;

message(0,"Klikk på et sted                    ") ->;
message(1,"Bjørnøya                            ") ->;
message(2,"Vest-Spitsbergen                    ") ->;
message(3,"???              ") ->;
message(4,"4               ") ->;
message(5,"5              ") ->;
message(6,"6              ") ->;
message(7,"7              ") ->;
message(8,"8              ") ->;
message(9,"9        ") ->;
message(10,"10    ") ->;
message(11,"11    ") ->;
message(12,"12     ") ->;
message(13,"13    ") ->;
message(14,"14           ") ->;

rect(14,<189,44>.<236,101>) ->;
rect(13,<131,86>.<187,119>) ->;
rect(12,<139,16>.<187,55>) ->;
rect(11,<75,103>.<103,136>) ->;
rect(10,<74,66>.<125,92>) ->;
rect(9,<78,10>.<139,26>) ->;
rect(8,<74,90>.<131,103>) ->;
rect(7,<125,55>.<189,86>) ->;
rect(6,<125,86>.<131,90>) ->;
rect(5,<74,26>.<139,68>) ->;
rect(4,<46,26>.<74,102>) ->;
rect(3,<28,65>.<46,91>) ->;
rect(2,<28,37>.<46,65>) ->;
rect(1,<10,51>.<28,84>) ->;

button-set1(<250,5>.<295,20>,"Edit",edit) ->;
button-set1(<310,5>.<355,20>,"Stop",stop) ->;
button-set1(<250,25>.<295,40>,"OK",ok) ->;
button-set1(<310,25>.<355,40>,"Retry",retry) ->;

zone(0) ->;

;End world: sted

"bezier-values(p1,p2,p3,p4) inneholder funksjonen!!"

do-bezier(X-Akse,Y-Akse,P1,P2,P3,P4) ->
    find-bezier(X-Akse,Y-Akse)
    rule(bezier-values(P1,P2,P3,P4),nil)
    suppress(1)
    set-window("graphic",0)
    set-window("console",1)
    output("console");

find-bezier(X-Akse,Y-Akse) ->
    set-window("graphic",1,30,50,420,300)
    set-window("console",0)
    output("graphic")
    gr-load("Svalbard:macpaints:funksjonsskjema")
```

```
    gr-draw-buttons(knappe-mengde1)
    gr-moveto(240,150)
    gr-text(3,10,3.nil)
    outm("X:")
    outm(X-Akse)
    outm(" Y:")
    outm(Y-Akse)
    gr-text(4,12,nil)
    bezier-interaction
    bez-knappe-klikk-test;
find-bezier ->
    set-window("graphic",0)
    set-window("console",1)
    output("console");

error-hook(x) -> find-rule(bezier-values) suppress(1) fail;

bez-knappe-klikk-test ->
    EraseRect(<0,180>.<400,300>)
    gr-moveto(50,200)
    melding(5,M5)
    outml(M5)
    gr-click(1,X,Y)
    gr-button-hit(knappe-mengde1,<x,y>,a)
    action(A);

action(edit) ->
    find-rule(bezier-values)
    suppress(1)
    edit;
action(stop) ->
    find-rule(bezier-values)
    suppress(1)
    set-window("graphic",0)
    set-window("console",1)
    output("console")
    block-exit(16);
action(retry) ->
    find-rule(bezier-values)
    suppress(1)
    find-bezier;
action(ok) -> /;

in-part(P,I) ->
    rect(I,R)
    gr-in-rect(P,R)
    /;
in-part(P,0) ->;

bezier-interaction ->
    gr-moveto(50,200)
    melding(1,M1)
    outml(M1)
    gr-clickr(1,X1,Y1)
    gr-moveto(X1,Y1)
    gr-line(1,1)
    EraseRect(<0,180>.<400,300>)
    gr-moveto(50,200)
    melding(4,M4)
    outml(M4)
    gr-clickr(1,X4,Y4)
    gr-moveto(X4,Y4)
    gr-line(1,1)
    EraseRect(<0,180>.<400,300>)
    gr-moveto(50,200)
    melding(2,M2)
    outml(M2)
    gr-clickr(1,X2,Y2)
    gr-moveto(X2,Y2)
    gr-line(1,1)
    EraseRect(<0,180>.<400,300>)
    melding(3,M3)
    gr-moveto(50,200)
    outml(M3)
    gr-clickr(1,X3,Y3)
    gr-moveto(X3,Y3)
    gr-line(1,1)
    graf-intern(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,P1,P2,P3,P4)
    assert(bezier-values(P1,P2,P3,P4),nil)
    draw-bezier(P1,P2,P3,P4)
    /;

draw-bezier(P1,P2,P3,P4) ->
    bez-draw01(P1,P2,P3,P4,0)
    enum(T,100)
    val(div(float(T),+1.0e+2),T')
    bez-draw01(P1,P2,P3,P4,T');

"X og Y er truncated i interngraf"
"Trenger ikke den første her, og heller ikke den andre"

bez-draw01(P1,P2,P3,P4,+1.0) ->
```

```
    bezier(P1,P2,P3,P4,+1.0,X,Y)
    intern-graf(X,Y,X',Y')
    gr-lineto(X',Y');
bez-draw01(<X,Y>,P2,P3,P4,0) ->
    intern-graf(X,Y,X',Y')
    gr-moveto(X',Y');
bez-draw01(P1,P2,P3,P4,T) ->
    bezier(P1,P2,P3,P4,T,X,Y)
    intern-graf(X,Y,X',Y')
    gr-lineto(X',Y')
    fail;

"Første alternativ gjelder startpunktet"
"Tredje regel gir presisjon. Her brukes +-0.02"

find-y-bez(<X,Y>,P2,P3,P4,+0.0,X1,X,Y) ->
    val(inf(X1,add(X,+5.0e-2)),1)
    val(inf(sub(X,+5.0e-2),X1),1);
find-y-bez(P1,P2,P3,<X,Y>,+1.0,X1,X,Y) ->
    val(inf(X1,add(X,+5.0e-2)),1)
    val(inf(sub(X,+5.0e-2),X1),1);
find-y-bez(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,T,X5,X,Y) ->
    bez-formel(X1,X2,X3,X4,T,X)
    val(inf(X5,add(X,+2.0e-2)),1)
    val(inf(sub(X,+2.0e-2),X5),1)
    bez-formel(Y1,Y2,Y3,Y4,T,Y)
    /;
find-y-bez(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,T,X5,X,Y) ->
    val(add(T,+1.0e-2),T-PlusEn)
    find-y-bez(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,T-PlusEn,X5,X,Y);

bezier(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,T,X,Y) ->
    bez-formel(X1,X2,X3,X4,T,X)
    bez-formel(Y1,Y2,Y3,Y4,T,Y);

bez-formel(S-t,S-td,S-1d,S-1,T,V) ->
    val(inf(T,+1.0e-2),0)
    val(sub(+1.0,T),E-nT)
    val(sub(T,+1.0),T-En)
    val(mul(E-nT,mul(E-nT,mul(E-nT,S-t))),V-En)
    val(mul(+3.0,mul(T,mul(T-En,mul(T-En,S-td)))),V-To)
    val(mul(+3.0,mul(T,mul(T,mul(E-nT,S-1d)))),V-Tre)
    val(mul(T,mul(T,mul(T,S-1))),V-Fire)
    val(add(V-En,add(V-To,add(V-Tre,V-Fire))),V);

intern-graf(X,Y,X'',Y'') ->
    val(add(mul(+1.0e+2,X),+3.0e+1),X')
    val(add(sub(+1.0e+2,mul(+1.0e+2,Y)),+3.0e+1),Y')
    val(trunc(X'),X'')
    val(trunc(Y'),Y'');

graf-intern(<X1,Y1>,<X2,Y2>,<X3,Y3>,<X4,Y4>,P1,P2,P3,P4) ->
    val(div(sub(float(X1),+3.0e+1),+1.0e+2),X1')
    val(div(sub(float(X2),+3.0e+1),+1.0e+2),X2')
    val(div(sub(float(X3),+3.0e+1),+1.0e+2),X3')
    val(div(sub(float(X4),+3.0e+1),+1.0e+2),X4')
    val(div(sub(+1.0e+2,sub(float(Y1),+3.0e+1)),+1.0e+2),Y1')
    val(div(sub(+1.0e+2,sub(float(Y2),+3.0e+1)),+1.0e+2),Y2')
    val(div(sub(+1.0e+2,sub(float(Y3),+3.0e+1)),+1.0e+2),Y3')
    val(div(sub(+1.0e+2,sub(float(Y4),+3.0e+1)),+1.0e+2),Y4')
    eq(p4,<X4',Y4'>)
    eq(p1,<X1',Y1'>)
    eq(p2,<X2',Y2'>)
    eq(p3,<X3',Y3'>);

InvertRect(R) -> gr-rect(3,R);

EraseRect(R) -> gr-rect(2,R);

repeat ->;
repeat -> repeat;

melding(1,"Vis startpunkt                  ") ->;
melding(2,"Vis utgangsbue fra startpunkt   ") ->;
melding(3,"Vis utgangsbue fra sluttpunkt   ") ->;
melding(4,"Vis sluttpunkt                  ") ->;
melding(5,"Trykk på en knapp for nytt valg ") ->;

knappe-mengdel(<250,5>.<295,20>,"Edit",edit) ->;
knappe-mengdel(<310,5>.<355,20>,"Stop",stop) ->;
knappe-mengdel(<250,25>.<295,40>,"Retry",retry) ->;
knappe-mengdel(<310,25>.<355,40>,"OK",ok) ->;

rect(funksjon,<31,31>.<129.129>) ->;

;End world: Normal

;
```