Arendse Bernth
Datalogisk Inst. Københavns Univ.
Sigurdsgade 41, DK-2200 København N

LOGIK ANVENDT TIL OVERSÆTTELSE AF JAPANSK.

På Datalogisk Institut ved Københavns Universitet eksperimenteres for tiden med at anvende logikprogrammering som hjælpemiddel ved automatiseret oversættelse fra japansk til engelsk og dansk. Logikprogrammeringssproget Prolog anvendes til syntaksanalyse af japansk, opbygning af en prædikatlogisk repræsentation af teksten, fortolkning af denne og ved hjælp af ordbøger oversættelse til engelsk eller dansk.

1. Bemærkninger om det japanske sprog.

Japansk er ikke beslægtet med noget andet sprog (måske bortset fra koreansk), og det har sin egen måde at beskrive grammatik på.
Indo-europæiske begreber passer i virkeligheden ikke særligt godt til japansk. På trods af dette vil beskrivelsen af det japanske sprog anvende indo-europæisk terminologi, da det formodes, at læseren er mest fortrolig med denne. Der er altså tale om et vist misforhold mellem beskrive-måde og det beskrevne. Endvidere er der af fremstillingsmæsssige grunde foretaget nogle simplificeringer, f.eks. i omtalen af partikler.

1.1 Sætningskonstruktion.

Principielt er enhver ordrækkefølge tilladt, blot verbet kommer til sidst. I de allerfleste tilfælde vil man dog se denne rækkefølge:

subjekt indirekte-objekt objekt verbum.

Subjektet udelades som regel, hvis det fremgår af sammenhængen. Kasus angives ved efterstillede partikler, også kaldet postpositioner. Blandt de vigtigste partikler kan nævnes:

"wa" el. "ga" (subjektspartikel),
"ni" (indir. objektspartikel),
"o" (objektspartikel).

Der er dog en vis forskel på "wa" og "ga", som det vil føre for vidt at komme ind på her.
Enhver underordnet sætning skal komme før den tilhørende hovedsætning. Da verbet altid står sidst i en sætning, vil verbet i en relativ sætning stå umiddelbart foran det substantiv, som det er relativt til. Der findes ikke relative pronominer på japansk.

Logik anvendt til oversættelse af japansk
Arendse Bernth
Proceedings of NODALIDA 1983, pages 29-36

1.2 Verber og adjektiver.

På japansk er der kun ringe forskel på verber og adjektiver. Verber og adjektiver bøjes ikke i køn, tal eller kasus, derimod bøjes begge i tid.

Verber (og adjektiver) bruges ofte uden et eksplicit subjekt. Subjektet afgøres da af sammenhængen. Både et verbum og et adjektiv kan stå alene og danne en sætning.

Der findes reelt kun to tider: præsens og præteritum. Derimod findes der mange former, der angiver stemninger, troværdighed o. a. Endvidere findes der forskellige grader af høflighed. Foruden specielt høfligt sprog findes der til hvert verbum/adjektiv en kort (mindre høflig) og en længere (mere høflig) form. Brugen af de lange og korte former afhænger af omstændighederne, men en hovedregel er, at man i almindeligt, neutralt talesprog anvender den korte form i underordnede sætninger, og den lange i hovedsætninger. I skriftsprog anvendes for det meste udelukkende de korte former.

1.3 Eksempler på nogle japanske sætninger.

Eksempel I:	Eksempel II:
人は花を見る	花を見る人は笑う
i kunreisiki-transkription:	
hito wa hana o miru	hana o miru hito wa warau
'subj.par' obj.par	
menneske blomst ser	blomst ser menneske smiler
(et menneske ser en blomst)	(et menneske, der ser en blomst, smiler)

Prædikatlogisk repræsentation for:

Eksempel I:

```
exists(X,(hito(X) & exists(Y,(hana(Y) & present(miru(X,Y))))))
```

Eksempel II:

```
exists(X,(hito(X) & (exists(Y,(hana(Y) & present(miru(X,Y)))))
                  & present(warau(X)))).
```

2. Prolog og sprogbehandling.

Prolog er baseret på en delmængde af 1.ordens prædikatkalkule (såkaldte definitte klausuler).
En definit klausul er et udtryk på formen:

C0 <- C1,C2,...,Cn

hvor C0 er konklusionen og de øvrige Cer er betingelser. Ci har formen:

p(t1,t2,...,tk)

hvor p er et prædikat og ti termer.
Prolog kan dels opfattes som et problembeskrivelsesværktøj (det deklarative aspekt) og dels som et program (det procedurale aspekt).
Logikprogrammering er deskriptiv i modsætning til traditionelle programmeringssprog, der er preskriptive.
Dette betyder, at har man beskrevet et problem, har man -stort set- også løst det. Dette giver sig tydeligt udtryk i syntaksanalyseopgaver. En grammatik udtrykt i traditionel BNF-notation skrives meget nemt om til Prolog -og så får man i tilgift et program, der kan foretage syntaksanalyse.
Denne Prolog-grammatik kan ved indføjelse af nogle ekstra variable returnere et resultat af syntaksanalysen. Vi udfører forsøg med at lade en Prolog-grammatik danne en prædikatlogisk model af forskellige sætningstyper. Denne logiske model forsøges anvendt som mellemsprog ved automatiseret oversættelse.
I et Prolog-program lægger man sig ikke på forhånd fast på forudsætninger og resultater. Dette skyldes, at Prolog er deskriptiv: programmet beskriver relationen mellem ind- og uddata, men fortæller ikke noget om, hvad der er forudsætninger (inddata) og hvad der er resultater (uddata).
Dette betyder, at et program kan bruges "begge veje", f.eks. kan det samme program både differentiere og integrere. På denne måde kan man oversætte fra et sprog til et andet ved hjælp af et analyseprogram for hvert sprog, samt en ordbog.
Dette princip forsøger vi at anvende ved oversættelse fra japansk til

dansk.
F. eks. kan man som inddata til den japanske del give:

    hito wa hana o miru

Dette giver som resultat den logiske repræsentation som beskrevet i afsnit 1.3. Lader man den danske del (incl. ordbog) behandle dette, fås:

    et menneske ser en blomst

2.1 Eksempel på japansk grammatik.

Udtrykt i en BNF notation:

```
<japsentence>       ::= <subjectnounphrase><verbphrase>

<subjectnounphrase>::= <relclause><nounphrase>
                       <subjectparticle><determiner>

<objectnounphrase> ::= <relclause><nounphrase>
                       <objectparticle><determiner>

<relclause>         ::= E | <verbphrase>

<nounphrase>        ::= <noun>
<noun>              ::= hito | hana
<subjectparticle>   ::= wa | ga
<objectparticle>    ::= o
<determiner>        ::= E | daredemo

<verbphrase>        ::= <intransitiveverb> |
                       <objectnounphrase><transitiveverb>

<intransitiveverb> ::= warau
<transitiveverb>    ::= miru
```

E angiver den tomme streng.

Den samme grammatik udtrykt i Prolog (variable begynder med stort):

```
japsentence(S,T)        <- subjectnounphrase(S,U),
                           verbphrase(U,T).

subjectnounphrase(S,T) <- relclause(S,U),
                           nounphrase(U,V),
                           subjectpar(V,W),
                           determiner(W,T).

objectnounphrase(S,T)  <- relclause(S,U),
                           nounphrase(U,V),
                           objectpar(V,W),
                           determiner(W,T).

relclause(S,S).
relclause(S,T)  <- verbphrase(S,T).
nounphrase(S,T) <- noun(S,T).

subjectpar(S,T) <- c(wa,S,T).
subjectpar(S,T) <- c(ga,S,T).
objectpar(S,T)  <- c(o,S,T).

determiner(S,S).
determiner(S,T) <- c(daredemo,S,T).

verbphrase(S,T) <- intransverb(S,T).
verbphrase(S,T) <- objectnounphrase(S,U),
                   transverb(U,T).

intransverb(S,T) <- c(warau,S,T).
transverb(S,T)   <- c(miru,S,T).

noun(S,T)        <- c(hito,S,T).
noun(S,T)        <- c(hana,S,T).

c(Word,[Word|S],S).
```

'S','T','U','V',og 'W' fungerer som en slags pegepinde i sætningen. Den første Prolog-regel siger (fortolket deklarativt):
"en 'japsentence' består af en 'subjectnounphrase' efterfulgt af en 'verbphrase'"
eller (fortolket proceduralt):
"der er en 'japsentence' fra S til T, hvis der er en 'subjectnounphrase' fra S til U og en 'verbphrase' fra U til T".
c(hito,S,T) betyder "ordet 'hito' befinder sig i positionen mellem S og T".
Man kan nu spørge:

```
japsentence([hito,ga,warau],[]).
```

Programmet vil så undersøge, om der er tale om en lovlig japansk sætning.

2.2 Semantik.

Ovenstående program kan kun foretage syntaksanalyse og svare ja/nej. For at gemme et semantisk resultat er det nødvendigt at indføje yderligere variable. Som eksempel på denne udvidelse kan vi vise, hvorledes reglerne for 'japsentence', 'noun' og 'determiner' kommer til at se ud:

```
japsentence(X,Logic,S,T) <- subjectnounphrase(X,Vp,Subj,S,U),
                            verbphrase(X,Vp,Logic,S,T).

noun(X,hito(X),S,T) <- c(hito,S,T).

determiner(X,P1,P2,all(X,(P1=>P2)),S,T) <- c(daredemo,S,T).

determiner(X,P1,P2,exists(X,(P1&P2)),S,S).
```

Variablen 'X' angiver det, vi fokuserer på, og 'Logic' returnerer den logiske repræsentation af sætningen.
Reglen for substantiver knytter et substantiv til det fokus ('X'), vi har. Således er semantikken for 'hito' ('mand'):

```
hito(X)
```

Som det ses, opbygges selve den logiske struktur i reglerne for 'determiner'.

Første regel for 'determiner' tager sig af 'daredemo' ('enhver'). Semantikken for 'enhver' bliver et alkvantoriseret udtryk:

```
all(X,(a(X) => b(X)))
```

Anden regel tager sig af 'en'/'et'. Japansk har hverken bestemte eller ubestemte artikler, ejheller bøjes substantiver i tal. I dette eksempel regnes et japansk substantiv for singularis.
Semantikken for 'en'/'et' bliver et eksistenskvantoriseret udtryk:

```
exists(X,(a(X) & b(X)))
```

3. Konklusion.

Som det fremgår af ovenstående eksempel er det relativt enkelt at lave et analyse-program i Prolog, når man har gjort sig grammatikken klar. Et af formålene med dette skrift er at gøre opmærksom på dette. Endvidere kunne man tænke sig en analysator for hvert af mange forskellige sprog; man kunne så -i hvert fald i teorien- foretage oversættelse mellem vilkårlige sprog ved hjælp af disse programmer. Dette skyldes det fælles mellemsprog.
Det skal dog nævnes, at Prolog har visse begrænsninger f.eks. med hensyn til klausulerne og den rækkefølge, de bliver udført i. Dette medfører, at det deklarative aspekt ikke er 100% løsrevet fra det procedurale aspekt. Denne begrænsning ligger ikke i logikprogrammering som sådan, men i det på nuværende tidspunkt mest tilgængelige logikprogrammeringssprog, nemlig Prolog. I det japanske femte generationsprojekt arbejdes der på en højere grad af parallellitet for at afhjælpe problemet med udførelsesrækkefølgen.
Det egentlige problem består i at finde et godt mellemsprog. Her er anvendt en prædikatlogisk model baseret på hypoteserne fremstillet i Pereira(1980).
Det er klart, at det er nemmere at lave en god repræsentation af en tekst, jo mere præcist og entydigt sproget er. Japansk er et meget vagt og upræcist sprog; faktisk bryder man sig slet ikke om at udtrykke sig præcist for ikke at støde modparten. Som eksempel kan nævnes, at substantiver, verber og lign. ikke bøjes i tal, og at subjektet for det meste udelades.
Man må gætte sig til meningen udfra sammenhængen. Det er klart, at disse ting vanskeliggør en automatisk oversættelse betydeligt. Det har bestemt heller ikke været muligt at tage højde for alt; formålet er også snarere at afprøve nogle teknikker på nogle begrænsede dele

af sprogene.
Endvidere er det naturligvis essentielt for en god oversættelse, at systemet har adgang til en beskrivelse af et relevant verdensbillede, hvilket også ville hjælpe til at fastlægge meningen. Dette er et ikke uvæsentligt problem, som vi dog ikke beskæftiger os med.

4. Litteratur.

Clocksin,W.F. & C.S. Mellish(1981): "Programming In Prolog". Springer.

Colmerauer,A.(1982): "An Interesting Subset of Natural Language". i Clark,K. & S-A. Taernlund (eds.): "Logic Programming". Akad.Press.

Dahl,V.(1979): "Logical Design of Deductive Natural Language Consultable Data Bases." Proceedings of Very Large Data Bases.

Koch,G.(1981): "Grammars and Predicate Calculus". Diku Report 81/16, Datalogisk Inst. ved Københavns Universitet.

Koch,G.(1983): "Stepwise Development of Logic Programmed Software Development Methods". Diku Report 83/5, Datalogisk Inst. ved Københavns Universitet.

Kowalski,R.A.(1979): "Logic for Problem Solving". North-Holland.

Miller,R.A.(1967): "The Japanese Language". University of Chigago.

Pereira,F.C.N. & D.H.D Warren(1980): "Definite Clause Grammars for Language Analysis a Survey of the Formalism And a Comparison with Augmented Transition Networks". Artificial Intellingence 13,3.