Gregers Koch
KøbenHavns Universitet

COMPUTATIONAL LINGUISTICS AND MATHEMATICAL LOGIC FROM A COMPUTER SCIENCE POINT OF VIEW

Let us look at a special type of context sensitive grammars, the so-called Definite Clause Grammars, or DCG, as described by Pereira and Warren (1980). The following question seems natural here:

- Can every context sensitive language be described by means of a definite clause grammar ?

The question must be answered in the affirmative, as can be seen by simulating Turing machines or better Markov algorithms by means of definite clause grammars. This means that the definite clause grammars have at least the same computational power as the grammars of Chomsky type 0. And hence theoretically speaking they can also handle languages of Chomsky type 1.

If we focus on the question which practical methods have been developed to handle context sensitive languages by means of definite clause grammars or similar formalisms then the situation is much more unclear. It seems to be a problem with a considerable actual research effort. We may see the occuring of several new grammar types as the results of exactly this tendency, for instance Gapping Grammars in (Dahl & Abramson 1984, Dahl 1984) and Puzzle Grammars (Sabatier 1984).

The next natural and fundamental question seems to be the following

- Does there exist any known problems within the languages that can be described by means of definite clause grammars ?

It should be answered in the affirmative, actually there exist many problems. As an example we may look at possibly the most famous problem, the coordination problem.

The text that we want to examine consists simply of the Danish sentence

- Peter ønsker ikke at spise et æble.
  (Peter does not want to eat an apple)

What should be the output or the result of the process corresponding to this sentence as input? It should be a sort of semantic representation providing a reasonable picture of the meaning of the statement. We tend to argue for the use of logical representations, and also here in the choice of semantic representation, so the notion logico-semantic representation seems to be justified.

Are we able to construct an automatic translation of an appropriate sublanguage into such a logico-semantic representation ? Yes, for example it can be done by a Prolog program like the one described in the appendix. (Presumably we wanted a DCG describing the same translation, but in the used version of Micro-Prolog there is no support of DCGs).

During such an execution the result variable obtains a value to be output like this:

```
(Not
  (Oensker Peter (Exist
      x
      ((x
          Is
          Aeble) And (Spise
          Peter
          x)))))
```


Computational linguistics and mathematical logic from a computer science point of view
Gregers Koch
Proceedings of NODALIDA 1985, pages 131-141

which corresponds to the following logical formula in a more traditional notation:

$\neg$ Ønsker(Peter,$\exists$ x(Æble(x) & Spiser(Peter,x))).

We notice that this seems to be a socalled intensional reading, and that Montague grammars here show themselves (Montague 1974). Perhaps we rather wanted a socalled extensional reading like the following formula

$\forall$ x(Æble(x) $\rightarrow$ $\neg$ Ønsker(Peter,Spiser(Peter,x))).

The problem here seems to be due to scope problems, and this is exactly the wanted example of the coordination problem.

Another reasonable question is: what do the simplest logico-semantic assignments (or representation transformations) look like ? That question seems to have a trivial answer, namely the constant assignment functions mapping everything into e.g. false. So we should probably modify the question a little:

- How can we characterize the simplest non-constant logico-semantic assignments ?

To give a reasonable answer to this question we should discuss further what does the term "simplest" mean in this context.

Let us suppose that simplicity involves Frege's principle for referential transparency.

Under the asumption of the principle of referential transparency we can argue that a particular assignment is the simplest possible one, viz. an assignment giving a mixture of lambda calculus expressions and predicate calculus expressions as the logico-semantic representation.

It seems justified to call this representation a denotational semantic representation, in the style of (Milne & Strachey 1976, Gordon 1979).

Before presenting this denotational semantic assignment function called Sem, it seems appropriate to look at yet another little grammar G2 and its treatment by means of definite clause grammars. (Figure 3)

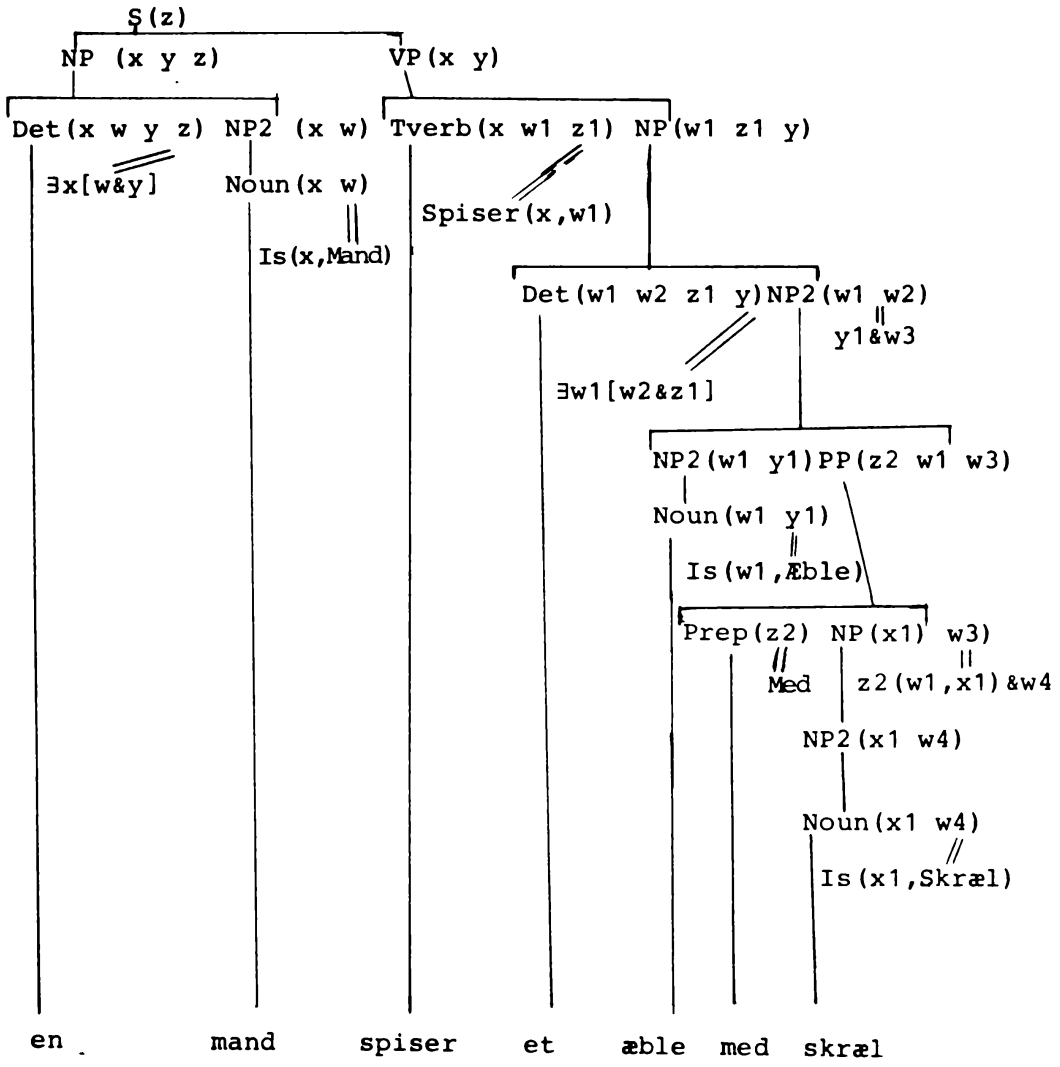
Perhaps we ought to show the system's treatment of a few selected sentences. The sentences are the following:

- En mand spiser et æble med skræl
  (a man eats an apple with the peel)

- Enhver ung mand elsker en smuk kvinde.
  (every young man loves a beatiful woman)

See the figures 1 and 2.

en mand spiser et æble med skræl



dvs.

$z = \exists x[w\&y]$

$= \exists x[w\&\exists w1[w2\&z1]]$

$= \exists x[w\&\exists w1[y1\&w3\&z1]]$

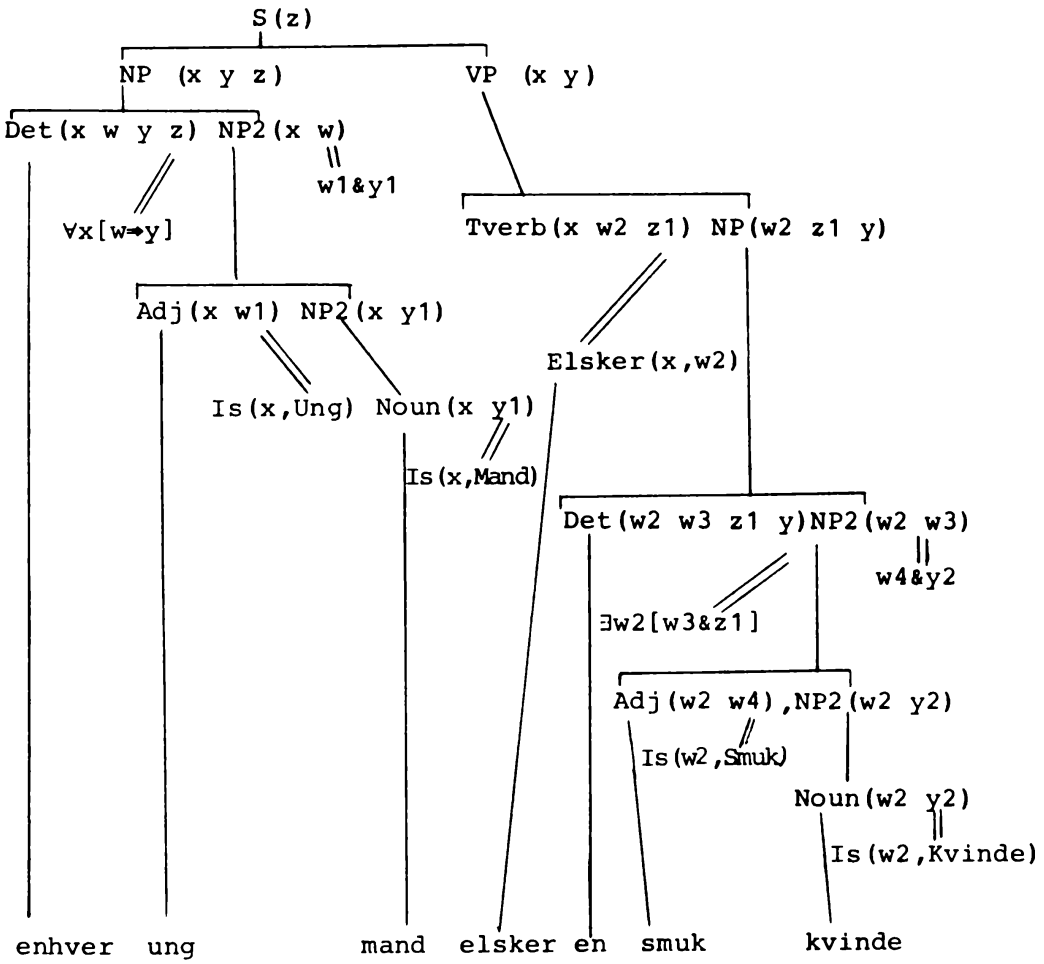
$= \exists x[Is(x,Mand)\&\exists w1[Is(w1,Æble)\&w3\&Spiser(x,w1)]]$

$= \exists x[Is(x,Mand)\&\exists w1[Is(w1,Æble)$
$\& z2(w1,x1)\&w4 \& Spiser(x,w1)]]$

$= \exists x[Is(x,Mand)\&\exists w1$
$[Is(w1,Æble)\&Med(w1,x1)\&Is(x1,Skræl)\&Spiser(x,w1)]]$

Figure 1

enhver ung mand elsker en smuk kvinde



dvs.

$z = \forall x[w \Rightarrow y]$

$= \forall x[w1 \& y1 \Rightarrow y]$

$= \forall x[w1 \& y1 \Rightarrow \exists w2[w3 \& z1]]$

$= \forall x[w1 \& y1 \Rightarrow \exists w2[w4 \& y2 \& z1]]$

$= \forall x[Is(x,Ung) \& Is(x,Mand) \Rightarrow \exists w2[Is(w2,Smuk) \& Is(w2,Kvinde) \& Elsker(x,w2)]]$

$= \forall x[Is(x,Ung) \& Is(x,Mand) \Rightarrow \exists [Is(y,Smuk) \& Is(y,Kvinde) \& Elsker(x,y)]]$

Figure 2

Now let us return to the problem to construct the simplest possible logico-semantic assignment function. We can make two assertions here:

Assertion A:
The following logico-semantic assignment function Sem in the grammar G2 is the simplest non-constant assignment function according to the mentioned criteria.

Assertion B:
The logico-semantic representation of a text by means of this denotational semantic assignment function Sem is identical to the corresponding logico-semantic representation of the text in the grammar G2 (which is the output corresponding to the result variable z when executing the definite clause grammar corresponding to G2).

The assertion A may be demonstrated to hold by always choosing the simplest possible alternatives.

The assertion B may be proven formally (though not possible within the limits of this paper), but the assertion may also be accepted by looking at a few examples. Let us try, using the example grammar G2 and the small texts above.

```
S    → NP     VP
NP   → Det    NP2
NP   → NP2
NP2  → Noun
NP2  → Adj    NP2
NP2  → NP2    PP
PP   → Prep   NP
VP   → IVerb
NP   → TVerb  NP
VP   → VP     PP
Det  → enhver
Det  → ethvert
Det  → en
Det  → et
Det  → hvilken
Det  → hvilket
```

Figure 3

Sem("en mand spiser et æble med skræl")

= Sem(en^mand^spiser^et^æble^med^skræl)

= Sem(D1^N1^Tv^D2^N2^P^Nart)

= $\text{Combine}_{[D,N,Tv,D,N,P,Nart]}$(Sem(D1),Sem(N1)Sem(Tv),Sem(D2),
Sem(N2),Sem(P),Sem(Nart))

= Sem(D1)(Sem(N1),λt.[Sem(D2)(Sem(N2)&&(Sem(P)oSem(Nart)),
First(Sem(Tv),t)])

= Sem(D1)(Sem(N1),λt.[Sem(D2)(Sem(N2)&&(Sem(P)oSem(Nart)),
First(λ(u,v).Tv(u,v),t))])

= Sem(D1)(Sem(N1),λt.[Sem(D2)(Sem(N2)&&(Sem(P)oSem(Nart)),
λv.Tv(t,v))])

= Sem(D1)(Sem(N1),λt.[(λ(u,w).∃y[u(y)&w(y)])
((λv.Is(v,N2))&&((λ(s,t).P(s,t))o(Nart)),λv.Tv(t,v))])

= Sem(D1)(Sem(N1)λt.[(λ(u,w).∃y[u(y)&w(y)])
((λv.Is(v,N2))&&(λs.P(s,Nart)),λv.Tv(t,v))])

= Sem(D1)(Sem(N1),λt.[(λ(u,w).∃y[u(y)&w(y)])
(λs.(Is(s,N2)&P(s,Nart)),λv.Tv(t,v))])

= Sem(D1)(Sem(N1),
λt.∃y[(λs.(Is(s,N2)&P(s,Nart)))(y)&(λv.Tv(t,v))(y)])

= Sem(D1)(Sem(N1),
λt.∃y[Is(y,N2)&P(y,Nart)&Tv(t,y)])

= (λ(u,v).∃x[u(x)&v(x)])
(λs.Is(s,N1),λt.∃y[Is(y,N2)&P(y,Nart)&Tv(t,y)])

= ∃x[(λs.Is(s,N1))(x)
&(λt.∃y[Is(y,N2)&P(y,Nart)&Tv(t,y)])(x)]

= ∃x(Is(x,N1)&∃y[Is(y,N2)&P(y,Nart)&Tv(x,y)]]

= ∃x(Is(x,Mand)&∃y[Is(y,Æble)&Med(y,Skræl)&Spiser(x,y)]].

```
  Sem("enhver ung mand elsker en smuk kvinde")
= Sem(enhver^ung^mand^elsker^en^smuk^kvinde)
= Sem(D1^A1^N1^Tv^D2^A2^N2)
= Combine[D,A,N,Tv,D,A,N](Sem(D1),Sem(A1),Sem(N1),Sem(Tv),
                          Sem(D2),Sem(A2),Sem(N2))

= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(Sem(A2)&&Sem(N2),First(Sem(Tv),t))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(λu.Is(u,A2)&&λv.Is(v,N2),
                              First(λ(x,y).Tv(x,y),t))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(Sem(D2)(λu.(Is(u,A2)&Is(u,N2)),λy.Tv(t,y))))
= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.((λ(x,z).∃v[x(v)&z(v)])(λu.(Is(u,A2)&Is(u,N2)),
                                        λy.Tv(t,y))))

= Sem(D1)(Sem(A1)&&Sem(N1),
        λt.(∃v[(λu.(Is(u,A2)&Is(u,N2)))(v)&(λy.Tv(t,y))(v)]))
= (λ(y,z).∀x[y(x)➡z(x)])(λu.Is(u,A1))&&(λv.Is(v,N1))),
        λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))
= (λ(y,z).∀x[y(x)➡z(x)])(λu.(Is(u,A1)&Is(u,N1)),
        λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))
= ∀x[(λu.(Is(u,A1)&Is(u,N1)))(x)
        ➡(λt.(∃v[Is(v,A2)&Is(v,N2)&Tv(t,v)]))(x)]
= ∀x[Is(x,A1)&Is(x,N1)➡∃v[Is(v,A2)&Is(v,N2)&Tv(x,v)]]


= ∀x[Is(x,Ung)&Is(x,Mand)
        ➡ ∃v[Is(v,Smuk)&Is(v,Kvinde)&Elsker(x,v)]]
```

References

Allwood, J. et al. "Logic in Linguistics", Cambridge University Press, 1977.

Cresswell, M. "Logics and Languages", Methuen, 1973.

Montague, R. "Formal Philosophy: Selected Papers of Richard Montague", ed. R. Thomason, Yale University Press, 1974.

Bernth, A. "Logic Programming and Translation Between Natural Languages", NordDATA Proceedings 3, 727-731, 1984, Helsinki.

Bernth, A. "Indføring i Prolog for Lingvister", noter, Datalogisk Institut ved Københavns Universitet, 1985.

Blikle, A. "Applied Denotational Semantics", lecture notes, Polish Academy of Sciences, Warsaw, 1985.

Blikle, A. & Tarlecki, A. "Naive Denotational Semantics", Proc. IFIP 1983, Paris, 345-356.

Charniak,E. & Wilks, Y.(eds.), "Computational Semantics", North-Holland, 1976.

Clocksin, W.F. & Mellish, C.S. "Programming in Prolog", Springer-Verlag, 1981.

Dahl, V. "More on Gapping Grammars", Proc.Fifth Gener. Comp. Systems 1984, Tokyo, p. 669 ff.

Dahl, V. & McCord, M. "Treating Coordination in Logic Grammars", Am. Journ. Comp. Ling. 1983.

Dahl, V. & Abramson, H. "On Gapping Grammars", Proc. Intern. Conf. Logic Programming, 1984, 77-88.

Gordon, M. J. C. "The Denotational Description of Programming Languages: An Introduction", Springer, 1979.

Hopcroft J.E. & Ullman, J.D. "Formal Languages and Their Relation to Automata", Addison-Wesley, 1969.

Koch, G. "Stepwise Development of Logic Programmed Software Development Methods", DIKU report 83/5, 1983.

Koch, G.(ed.), "Fifth Generation Programming vol. 1: Logic Programming in Natural Language Analysis", Proceedings of Workshop in Copenhagen, Dec. 1984, DIKU report 85/2.

Koch, G. "Who is a Fallible Greek in Logic Grammars", p.54-77 in (Koch 85 a), 1985 b.

Milne, R.E. & Strachey, C."A Theory of Programming Language Semantics", Chapman and Hall, 1976.

Pereira F. & Warren, D. "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, 13,3 (1980), 231-278.

Sabatier, P. "Puzzle Grammars", Proc. Natural Language Understanding and Logic Programming, Rennes, 1984, 121 - 136.

Sabatier, P. "Des Grammaires de Metamophose aux Grammaires de Puzzle", Copenhagen Workshop on Logic Programming in Natural Language Analysis, 1984.

Saint-Dizier, P. "On Syntax and Semantics of Modifiers in Natural Language Sentences" in (Koch 85 a), 1985.

Appendix

```
Dt (X Y Z) if
          Ds (X Y Z)
Dt ((Which X (Y And Z)) x y) if
          c (hvilken x z) and
          Dn (X Y z X1) and
          Dvp (X Z X1 y)
Dt ((Which X (Y And Z)) x y) if
          c (hvilket x z) and
          Dn (X Y z X1) and
          Dvp (X Z X1 y)
Ds1 (X Y Z) if
          Dnp (x y X Y z) and
          Dvp (x y z Z)
Ds ((X And Y) Z x) if
          Ds1 (X Z y) and
          c (og y z) and
          Ds (Y z x)
Ds (X Y Z) if
          Ds1 (X Y Z)
Ds ((X Imp Y) Z x) if
          c (hvis Z y) and
          Ds (X y z) and
          c (saa z X1) and
          c (gaelder X1 Y1) and
          c (at Y1 Z1) and
          Ds (Y Z1 x)
Dnp (X Y Z x y) if
          Dd (X z Y Z x X1) and
          Dn (X Y1 X1 Z1) and
          Drc (X Y1 z Z1 y)
Dnp (X Y Y Z x) if
          Dprop (X Z x)
Drc (X Y Y Z Z)
Drc (X Y (Y And Z) x y) if
          c (der x z) and
          Dvp (X Z z y)
Drc (X Y (Y And (Not Z)) x y) if
          c (der x z) and
          c (ikke z X1) and
          Dvp (X Z X1 y)
Drc (X Y (Y And Z) x y) if
          c (som x z) and
          Dnp (X1 Y1 Z z Z1) and
          Dtv (X1 X Y1 Z1 y)
Drc (X Y (Y And (Not Z)) x y) if
          c (som x z) and
          Dnp (X1 Y1 Z z Z1) and
          c (ikke Z1 x1) and
          Dtv (X1 X Y1 x1 y)
Dvp1 (X (Not Y) Z x) if
          Dnegvp (X Y Z x)
Dvp1 (X Y Z x) if
          Dposvp (X Y Z x)
```

```
Dvp (X (Y And Z) x y) if
          Dvp1 (X Y x z) and
          c (og z X1) and
          Dvp (X Z X1 y)
Dvp (X Y Z x) if
          Dvp1 (X Y Z x)
Dnegvp (X Y Z x) if
          Dvp-vp (X y Y Z z) and
          c (ikke z X1) and
          c (at X1 Y1) and
          Di (X y Y1 x)
Dnegvp (X Y Z x) if
          Dtv (X y z Z X1) and
          c (ikke X1 Y1) and
          Dnp (y z Y Y1 x)
Dnegvp (X Y Z x) if
          Div (X Y Z y) and
          c (ikke y x)
Dposvp (X Y Z x) if
          Dvp-vp (X y Y Z z) and
          c (at z X1) and
          Di (X y X1 x)
Dposvp (X Y Z x) if
          Dtv (X y z Z X1) and
          Dnp (y z Y X1 x)
Dposvp (X Y Z x) if
          Div (X Y Z x)
Di (X Y Z x) if
          Dti (X y z Z X1) and
          Dnp (y z Y X1 x)
Di (X Y Z x) if
          Dii (X Y Z x)
Di (X Y Z x) if
          Dvp-vpi (X y Y Z z) and
          c (at z X1) and
          Di (X y X1 x)
Dd (X Y Z (All X (Y Imp Z)) x y) if
          c (enhver x y)
Dd (X Y Z (All X (Y Imp Z)) x y) if
          c (ethvert x y)
Dd (X Y Z (Exist X (Y And Z)) x y) if
          c (en x y)
Dd (X Y Z (Exist X (Y And Z)) x y) if
          c (et x y)
Div (X (Y X) Z x) if
          c (Y Z x) and
          Diverbs (y) and
          Y Member y
Dtv (X Y (Z X Y) x y) if
          c (Z x y) and
          Dtverbs (z) and
          Z Member z
```

```
Dvp-vp (X Y (Z X Y) x y) if
         c (Z x y) and
         Dvp-vps (z) and
         Z Member z
Dii (X (Y X) Z x) if
         c (Y Z x) and
         Diinfs (y) and
         Y Member y
Dti (X Y (Z X Y) x y) if
         c (Z x y) and
         Dtinfs (z) and
         Z Member z
Dvp-vpi (X Y (Z X Y) x y) if
         c (Z x y) and
         Dvp-vpinfs (z) and
         Z Member z
Dn (X (X Is Y) Z x) if
         c (Y Z x) and
         Dnouns (y) and
         Y Member y
Dprop (X Y Z) if
         c (X Y Z) and
         Dprops (x) and
         X Member x
Diverbs ((lever blomstrer smiler flyver))
Dtverbs ((elsker spiser er har holder-af bestiger))
Dvp-vps ((oensker haaber))
Diinfs ((leve blomstre smile flyve))
Dtínfs ((elske spise vaere have holde-af bestige))
Dvp-vpinfs ((oenske haabe))
Dnouns ((kvinde aeble mand soft-ice fugl trae medlem skisportsmand...
 menneske ting))
Dprops ((Mary John Peter Prolog Tony Mike sne regn))
```