6 Strojový překlad a Prolog

6.1 Logická forma jako interlingva

příklad	logická reprezentace	zápis v Prologu
Felix	$felix$	`felix`
kocour	$\lambda x(cat(x))$	`X^cat(X)`
velký	$\lambda x(big(x))$	`X^big(X)`
velký kocour	$\lambda x(big(x) \wedge cat(x))$	`X^(big(X),cat(X))`
mňoukal	$\lambda x(meowed(x))$	`X^meowed(X)`
honil Felixe	$\lambda x(chased(x, felix))$	`X^chased(X,felix)`
honil	$\lambda y \lambda x(chased(x, y))$	`Y^X^chased(X,Y)`
je kocour	$\lambda x(cat(x))$	`X^cat(X)`
je velký	$\lambda x(big(x))$	`X^big(X)`
bez Felixe	$\lambda x(without(x, felix))$	`X^without(X,felix)`
bez	$\lambda y \lambda x(without(x, y))$	`Y^X^without(X,Y)`
Felix mňoukal.	$meowed(felix)$	`meowed(felix)`
(Nějaký) kocour mňoukal.	$\exists x(cat(x) \wedge meowed(x))$	`some(X,cat(X),meowed(X))`
Každý kocour mňoukal.	$\forall x(cat(x) \rightarrow meowed(x))$	`all(X,cat(X),meowed(X))`
Fido honil kočku.	$\exists x(cat(x) \wedge chased(fido, x))$	`some(X,cat(X),chased(fido,X))`
Každý pes honil kočku.	$\forall x(dog(x) \rightarrow \exists y(cat(y) \wedge chased(x, y)))$	`all(X,dog(X),some(Y,cat(Y),chased(X,Y)))`
(N.) pes honil všechny kočky.	$\exists x(dog(X) \wedge \forall y(cat(y) \rightarrow chased(x, y)))$	`some(X,dog(X),all(Y,cat(Y),chased(X,Y)))`
(nějaký) kocour	$\lambda Sco \exists x(cat(x) \wedge Sco(x))$	`(X^Sco)^some(X,cat(X),Sco)`
(nějaký)	$\lambda Res \lambda Sco \exists x(Res(x) \wedge Sco(x))$	`(X^Res)^(X^Sco)^some(X,Res,Sco)`
každý kocour	$\lambda Sco \forall x(cat(x) \rightarrow Sco(x))$	`(X^Sco)^all(X,cat(X),Sco)`
každý	$\lambda Res \lambda Sco \forall x(Res(x) \rightarrow Sco(x))$	`(X^Res)^(X^Sco)^all(X,Res,Sco)`

```
% English <-> Latin interlingual translation
% from M.A.Covington: Natural Language Processing for Prolog Programmers,
% Prentice Hall 1994, p.234-238

% Control

e_l(E,L) :- es(Sem,E,[]), ls(Sem,L,[]).
e_l(E,L,Sem) :- es(Sem,E,[]), ls(Sem,L,[]).

% English

es(Sem) --> enp((X^Sco)^Sem), evp(X^Sco).

enp(Sem) --> ed((X^Res)^Sem), en(X^Res).

evp(Sem) --> ev(Sem).
evp(X^Pred) --> ev(Y^X^Sco), enp((Y^Sco)^Pred).

ed((X^Res)^(X^Sco)^all(X,Res,Sco)) --> [every].
ed((X^Res)^(X^Sco)^some(X,Res,Sco)) --> [a]; [some].

en(X^dog(X)) --> [dog].
en(X^cat(X)) --> [cat].

ev(X^meowed(X)) --> [meowed].
ev(Y^X^chased(X,Y)) --> [chased].
ev(Y^X^saw(X,Y)) --> [saw].
```

```
% Latin

ls(Sem) --> lnp(_,nom,(X^Sco)^Sem), lvp(X^Sco).

lvp(Sem) -->    lv(Sem).
lvp(X^Pred) --> lnp(_,acc,(Y^Sco)^Pred), lv(Y^X^Sco).

lnp(Gender,Case,Sem) -->
  ld(Gender,Case,(X^Res)^Sem), ln(Gender,Case,X^Res).
lnp(Gender,Case,Sem) -->
  ln(Gender,Case,X^Res), ld2(Gender,Case,(X^Res)^Sem).

ld(_,_,  (X^Res)^(X^Sco)^some(X,Res,Sco)) --> [].
ld(_,nom,(X^Res)^(X^Sco)^all(X,Res,Sco))  --> [omnis].
ld(_,acc,(X^Res)^(X^Sco)^all(X,Res,Sco))  --> [omnem].

ld2(masc,nom, (X^Res)^(X^Sco)^some(X,Res,Sco)) --> [quidam].
ld2(masc,acc, (X^Res)^(X^Sco)^some(X,Res,Sco)) --> [quendam].
ld2(fem, nom, (X^Res)^(X^Sco)^some(X,Res,Sco)) --> [quaedam].
ld2(fem, acc, (X^Res)^(X^Sco)^some(X,Res,Sco)) --> [quandam].

ln(masc,nom, X^dog(X)) --> [canis].
ln(masc,acc, X^dog(X)) --> [canem].
ln(fem, nom, X^cat(X)) --> [felis].
ln(fem, acc, X^cat(X)) --> [felem].

lv(X^meowed(X))     --> [ululavit].
lv(Y^X^chased(X,Y)) --> [agitavit].
lv(Y^X^saw(X,Y))    --> [vidit].
```

6.2 Transfer na úrovni závislostní reprezentace

```
% English <-> Czech transfer translation
% with a quasi-dependency syntactic representation

% Fancy input and output routines

:- reconsult('output.pl'), reconsult('input.pl').

% only input and output

ec :- read_in(Text),e_c(Text,Result),write_word(Result).
ce :- read_in(Text),c_e(Text,Result),write_word(Result).

% input, output and analyses

ecs :- read_in(Text),e_c(Text,Result,ES,CS),
       write_word(Result),nl,pp(ES),nl,nl,pp(CS),nl.
ces :- read_in(Text),c_e(Text,Result,CS,ES),
       write_word(Result),nl,pp(CS),nl,nl,pp(ES),nl.


% Control

e_c(EText,CText) :-
  es(EStr,EText,[.]),
  x(EStr,CStr),
  cs(CStr,CText,[.]).

e_c(EText,CText,EStr,CStr) :-
  es(EStr,EText,[.]),
  x(EStr,CStr),
  cs(CStr,CText,[.]).


c_e(CText,EText) :-
  cs(CStr,CText,[.]),
  x(EStr,CStr),
  es(EStr,EText,[.]).

c_e(CText,EText,CStr,EStr) :-
  cs(CStr,CText,[.]),
  x(EStr,CStr),
  es(EStr,EText,[.]).

% English

es(v(VLex,[n(NLex,subj)|Deps])) -->
  enp(n(NLex)), evp(v(VLex,Deps)).
enp(N) --> ed(_Det), en(N).
enp(PN) --> epn(PN).
evp(v(VLex,[n(NLex,obj)|Deps])) -->
  etv(v(VLex,Deps)), enp(n(NLex)).
evp(IV) --> eiv(IV).
```

```
epn(n(felix)) --> [felix].
epn(n(fido)) --> [fido].
en(n(dog)) --> [dog].
eiv(v(bark,[])) --> [barks].
eiv(v(meow,[])) --> [meows].
etv(v(chase,[])) --> [chases].
ed(d(the)) --> [the].
ed(d(a)) --> [a].

% Czech

cs(v(VLex,[n(NLex,subj)|Deps])) -->
  cn_nom(n(NLex)), cvp(v(VLex,Deps)).
cvp(v(VLex,[n(NLex,obj)|Deps])) -->
  ctv(v(VLex,Deps)), cn_acc(n(NLex)).
cvp(IV) --> civ(IV).

cn_nom(n(mikeš)) --> [mikeš].
cn_nom(n(fousek)) --> [fousek].
cn_nom(n(pes)) --> [pes].
cn_acc(n(mikeš)) --> [mikeše].
cn_acc(n(fousek)) --> [fouska].
cn_acc(n(pes)) --> [psa].
civ(v(štěkat,[])) --> [štěká].
civ(v(mňouká,[])) --> [mňouká].
ctv(v(honit,[])) --> [honí].


% Transfer

xlex(dog,pes).
xlex(felix,mikeš).
xlex(fido,fousek).
xlex(chase,honit).
xlex(bark,štěkat).
xlex(meow,mňoukat).


x(v(SVLex,SDeps),v(TVLex,TDeps)) :-
  xlex(SVLex,TVLex),
  xdeps(SDeps,TDeps).

xdeps([],[]).
xdeps([n(SLex,Fun)|SDeps],[n(TLex,Fun)|TDeps]) :-
  xlex(SLex,TLex),
  xdeps(SDeps,TDeps).
```