SINTAKSA JEZIKOV IN GRAMATIKE

Ivan Bratko
Univerza v Ljubljani

JEZIKI IN STAVKI

- Jezik = množica nizov (stavkov)

- Primeri stavkov in jezikov:

  Zelo abstrakten jezik:
  - ab, aabb, aaabbb, ...

  Programski jezik:
  - x = a + b
  - if x < y then z = x else z = y

  Naravni jezik:
  - John paints.
  - The cat scares a mouse.

DEFINICIJA JEZIKOV

- Definiramo sintakso in semantiko (pomen)
- Definicija sintakse navadno z *gramatiko*
- Definicija semantike, razni načini:
  - prevajalska semantika,
  - operacijska semantika
  - denotacijska
  - atributna gramatika
  - aksiomatska
  - ...

DEFINIRANJE SINTAKSE Z GRAMATIKO (S PREPISNIMI PRAVILI)

Primer:

s → a b

s → a s b

s neterminalni simbol

a, b terminalna simbola

s začetni simbol

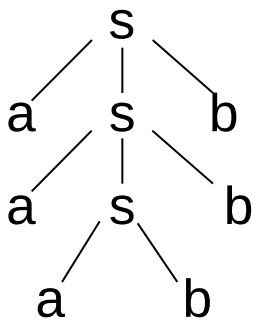
Krajši zapis:

s → a b | a s b

GRAMATIKA GENERIRA STAVKE JEZIKA

s → a b

s → a s b



Generiranje od zgoraj navzdol

Generirani stavek (niz) = a a a b b b

Generirani stavek vsebuje samo terminalne simbole

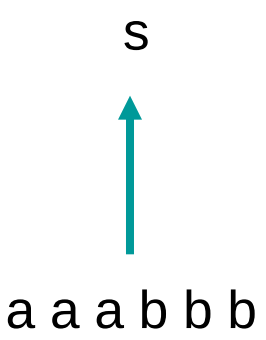
GRAMATIKA TUDI RAZPOZNAVA STAVKE JEZIKA

s → a b

s → a s b

Npr. dan je stavek a a a b b b

Ali stavek pripada jeziku?

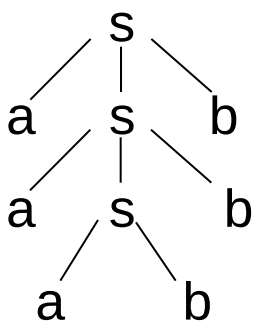


Razpoznavanje: od spodaj navzgor

SINTAKSNA RAZGRADNJA
(SYNTACTIC PARSING)

Dani stavek: aaabbb

Sintaksna struktura stavka (sintaksno drevo, angl. parse tree):



BNF ZAPIS GRAMATIK
BACKUS-NAUR FORM

<s> ::= a b | a <s> b

Koničasti oklepaji označujejo neterminalne simbole

IZRAZI V PROGRAMSKEM JEZIKU

<exp> ::= a | b | c | ...
<exp> ::= <exp> + <exp> | <exp> * <exp>
<exp> ::= (<exp>)

Stavki tega jezika:

a
a + b
a + b * c
(a + b) * (a + c)
((a + b) * c)
...

Pazi: Ta gramatika lahko razgradi stavke na razne načine (*dvoumnost*)!

Gramatika je dvoumna (nezaželeno, dvoumen pomen)

ZAPOREDJA UKAZOV ZA ROBOTA

up

up up down up down

USTREZNA BNF GRAMATIKA

<move> ::= <step> | <step> <move>

<step> ::= up | down

DCG GRAMATIKE

- DCG, Definite Clause Grammars
- *Definite Clause* = stavek v prologu (vendar ne vprašanje)
- Te gramatike so navadno vgrajene v prolog
- Prolog sam je že implementacija DCG gramatik
- Generira stavke in razgrajuje stavke

DCG ZA ROBOTOVE PROGRAME

```
move  -->  step.
move  -->  step, move.
step  -->  [up].
step  -->  [down].
```

PREDSTAVITEV STAVKOV V DCG

DCG pričakuje stavke kot „diferenčne sezname“ (difference list)

aabb ~ [a,a,b,b], [ ]
~ [a,a,b,b,c], [c]
~ [a,a,b,b,x,y,z], [x,y,z]
~ ...

up up down up ~ [up, up, down, up], [ ]
~ [up, up, down, up, left, right], [left, right]
~ [up, up, down, up | L], L

VPRAŠANJA ZA DCG

?- move([up,down,up], []).
yes

?- move([up, X, up], []).
X = up;
X = down;
no

?- move([up, down, up, a, b, c], [a, b, c]).
yes

PROLOG PREVEDE DCG V ANALIZNI PROGRAM

```
move( List, Rest)  :-
    step( List, Rest).

move( List1, Rest)  :-
    step( List1, List2),
    move( List2, Rest).

step( [up | Rest], Rest).

step( [down | Rest], Rest).
```

PREVOD DCG V PROLOG

DCG rule:

```
move  -->  step, move.
```

Corresponding Prolog rule:

```
move( List1, Rest)  :-
    step( List1, List2),
    move( List2, Rest).
```

DIFERENČNI SEZNAMI

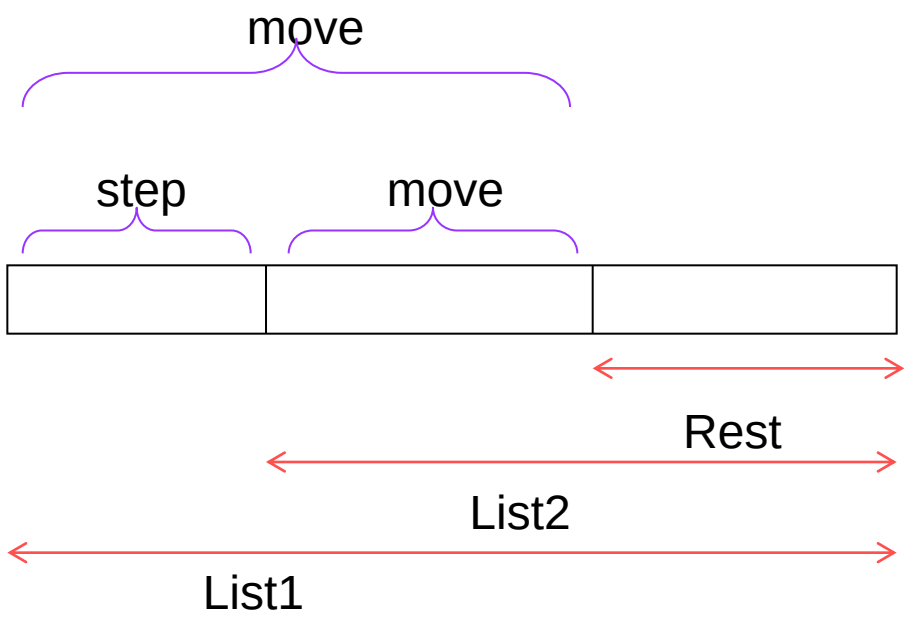
- Omogočajo učinkovito konkatenacijo seznamov
- Deluje podobno kot kazalec na konec seznama
- Sznam L predstavljen s parom A - Z

  A - Z predstavlja razliko med seznamoma A in Z
- Konkatenacija seznamov s konstantno časovno zahtevnostjo

  conc(A1 - Z1, Z1 - Z2, A1 - Z2).

PREVOD DCG V PROLOG

```
move( List1, Rest)  :-
    step( List1, List2),
    move( List2, Rest).
```

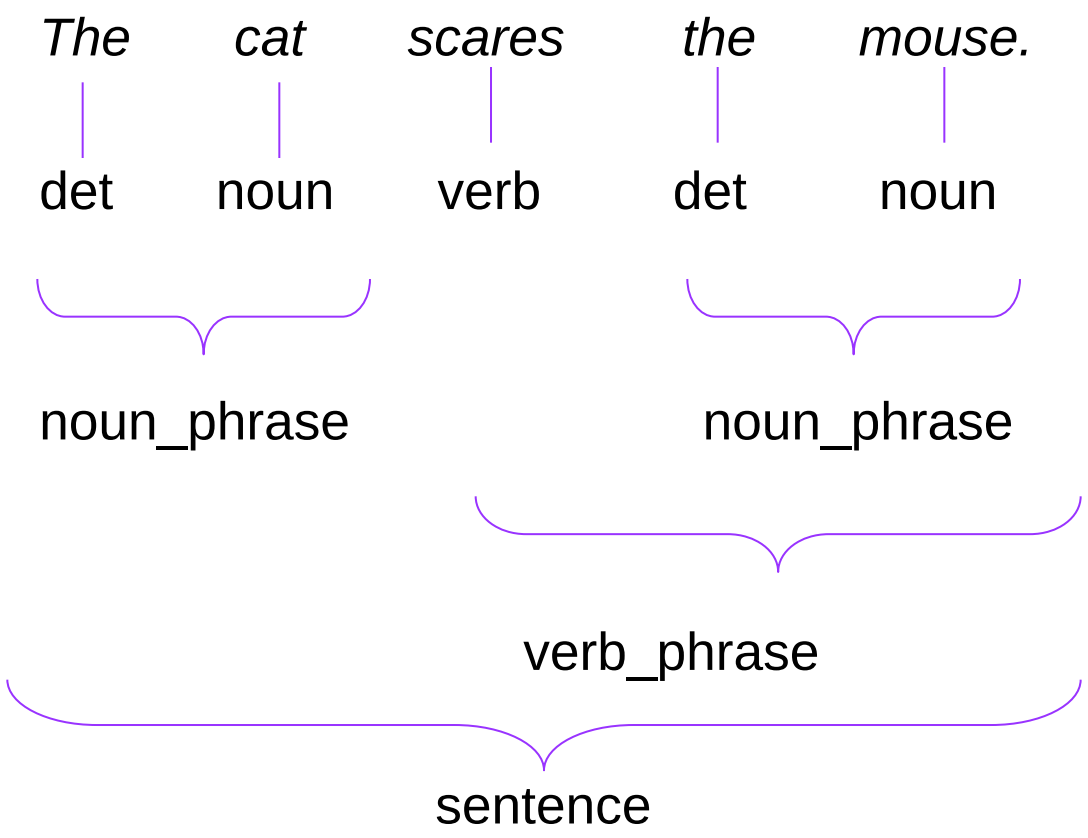


NALOGA

Prevedi v standardni Prolog:

```
s  -->  [a], s, [b].
```

ENOSTAVNA DCG GRAMATIKA ZA ANGL. JEZIK



ENOSTAVNA GRAMATIKA ZA ANGL.

```
sentence  -->  noun_phrase, verb_phrase.
verb_phrase  -->  verb, noun_phrase.
noun_phrase  -->  determiner, noun.
determiner  -->  [ the].
noun  -->  [ cat].
noun  -->  [ cats].
noun  -->  [ mouse].
verb  -->  [ scares].
verb  -->  [ scare].
```

TA GRAMATIKA GENERIRA STAVKE

[the, cat, scares, the, mouse]

[the, mouse, scares, the, mouse]

[the, cats, scare, the, mouse]

[the, cats, scares, the, mouse]

KONTEKSTNA ODVISNOST!

UJEMANJE V ŠTEVILU LAHKO VSILIMO Z ARGUMENTI

```
sentence( Number)  -->
      noun_phrase( Number), verb_phrase( Number).

verb_phrase( Number)  -->
      verb( Number), noun_phrase( Number1).

noun_phrase( Number)  -->
      determiner( Number), noun( Number).

noun( singular)  -->  [ mouse].
noun( plural)  -->  [ mice].
verb( singular)  -->  [scares].
verb( plural)  -->  [scare].
```

UJEMANJE V ŠTEVILU LAHKO VSILIMO Z ARGUMENTI

?- sentence(Number, [the, cats, scares, the, mouse], []).

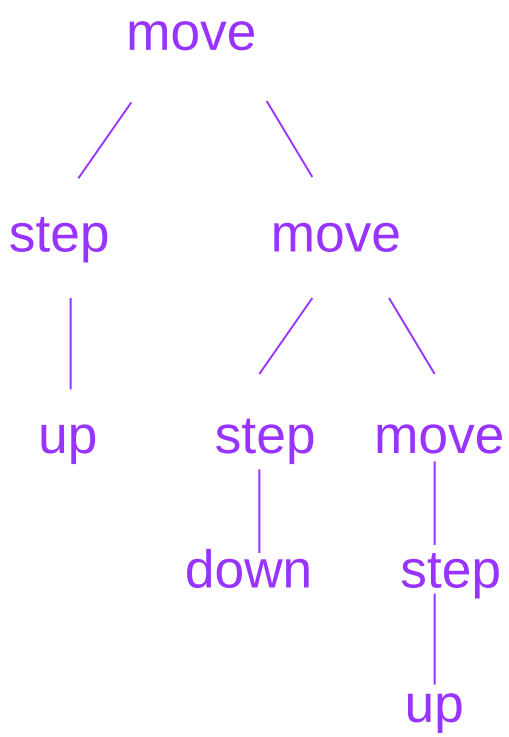
no

?- sentence(Number, [the, cats, scare, the, mouse], []).

Number = plural

KONSTRUIRANJE SINTAKSNIH DREVES

[up, down, up]



DCG, KI GRADI SINTAKSNA DREVESA

Tree = move(step(up), move(step(down), ...))

```
move( move( Step))  -->  step( Step).

move( move( Step, Move))  -->
      step( Step), move( Move).

step( step( up))  -->  [up].

step( step( down))  -->  [down].
```

DEFINIRANJE POMENA

N.pr.: Pomen od „move“ = Razdalja med začetno in končno točko

meaning (‘up up down up’) = 1+1-1+1 = 2

meaning(‘up up down up’) =

meaning(‘up’) + meaning(‘up down up’)

OD SINTAKSNEGA DREVESA DO POMENA

Določitev pomena stavka:

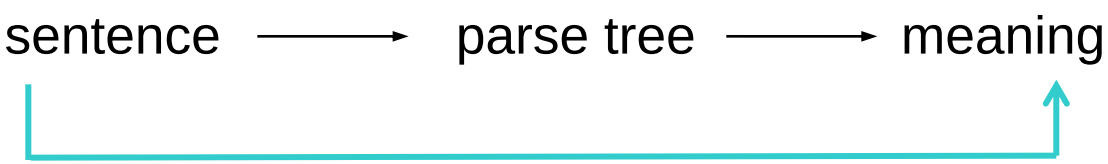
(1) Zgradi sintaksno drevo

(2) Izlušči pomen iz sintaksnega drevesa

POMEN SINTAKSNIH DREVES ZA „MOVE“

```
meaning( move( Step, Move), Dist)  :-
        meaning( Step, D1),
        meaning( Move, D2),
        Dist  is  D1 + D2.

meaning( move( Step), D)  :-
        meaning( Step, D).

meaning( step( up), 1).
meaning( step( down), -1).
```

DRUGA MOŽNOST: PREPLETEMO SINTAKSO IN SEMANTIKO



Sintaksno drevo zaobidemo, pomen vgrajen direktno v gramatiko

PROLOGOVI CILJI V DCG: {Goal}

```
move( Dist)  -->  step( Dist).

move( Dist)  -->
    step( D1), move( D2), {Dist  is  D1 + D2}.

step( 1)  -->  [ up].

step( -1)  -->  [ down].

?-  move( D, [ up, up, down, up], [ ] ).
     D  =  2
```

NALOGA

Naj bo dana naslednja gramatika:

```
move( X0, X)  -->
    step( Dx), { X is X0 + Dx, safe( X) }.

move( X0, X)  -->
   step( Dx), { X1 is X0 + Dx, safe( X1) }, move( X1, X).

step( 1)  -->  [right].
step( -1)  -->  [left].
safe( X)  :-
  -5 =< X, X =< 5.
```

NALOGA, NAD.

Kaj odgovori prolog na vprašanja:

(a) ?- move(2, Xa, [left,right,right,right], [ ]).

(b) ?- move(4, Xb, [left,right,right,right], [ ]).

(c) ?- conc(L, _ , _), move(5, 5 ,L, [ ]). % conc is list concatenation

(Podaj prve tri prologove odgovore)

(d) Naj gramatika definira gibanje robota. Kaj pomenita X in Y v move(X,Y)? Intuitivno, kakšne premike robota dovoljuje ta gramatika?

NALOGA, NAD.

- (e) Ali lahko gramatiko uporabimo v obratni smeri: podamo „pomen“ stavka, kaj je stavek s tem pomenom? Npr. program za robota, ki premakne robota iz 4 v -3. Postavi ustrezno vprašanje.

NALOGA

Dana je naslednja gramatika:

```
s( N, Dir) --> num(N).
s( N1, inc) --> num(N1), s(N2, inc), {N1 =< N2}.
s( N1, dec) --> num(N1), s(N2, dec), {N1 >= N2}.
num(N) --> [N], { N=1; N=2; N=3}.
```

Kaj prolog odgovori na naslednja vprašanja?

```
(a)  ?-  s( N, Da, [2,3], [ ]).
(b)  ?-  s( N, Db, [2, 3, 4], [ ]).
(c)  ?-  s( N, Dc, [ 1, 3, 2], [ ]).
(d)  ?-  s( N, Dd, [2, 2, 2, 2]).     % Give all Prolog's answers
(e)  ?-  s( 2, De, [ N1, N2], [ ]).  % Give all Prolog's answers
```

POMEN NARAVNEGA JEZIKA

- Kako predstaviti pomen stavkov naravnega jezika = ?

- Ena možnost je logika

PRIMERI POMENOV V LOGIKI

Stavek	Formalizirani pomen
``John paints''	*paints(john)*
``John likes Annie''	*likes(john, annie)*

PRIMERI POMENOV V LOGIKI

Stavek:

``A man paints''

Pomen:

exists(X, man(X) and paints(X))

Opomba: ``paints'' je netranzitivni glagol, ``likes'' je tranzitivni glagol

MOŽNA SINTAKSA

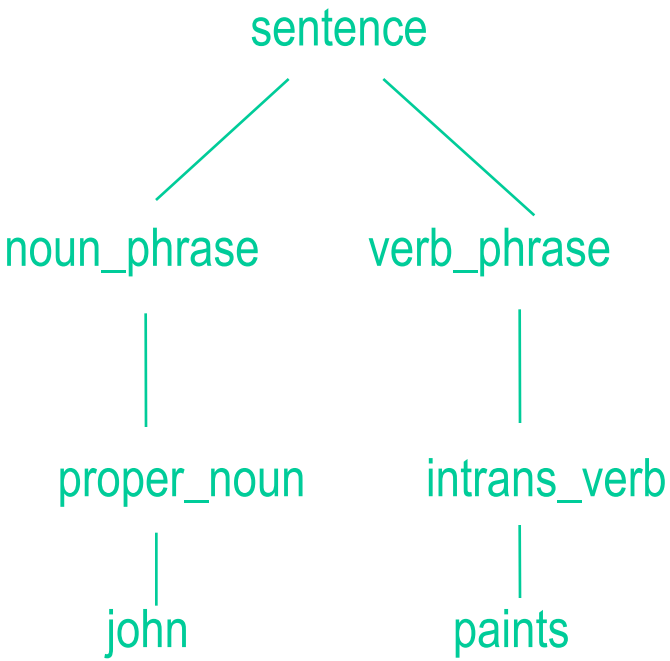
```
sentence  --->  noun_phrase,  verb_phrase.
noun_phrase  -->  proper_noun.
verb_phrase  -->  intrans_verb.
verb_phrase  -->  trans_verb, noun_phrase.
intrans_verb  -->  [ paints].
trans_verb  -->  [ likes].
proper_noun  -->  [ john].
...
```

VGRADNJA POMENA

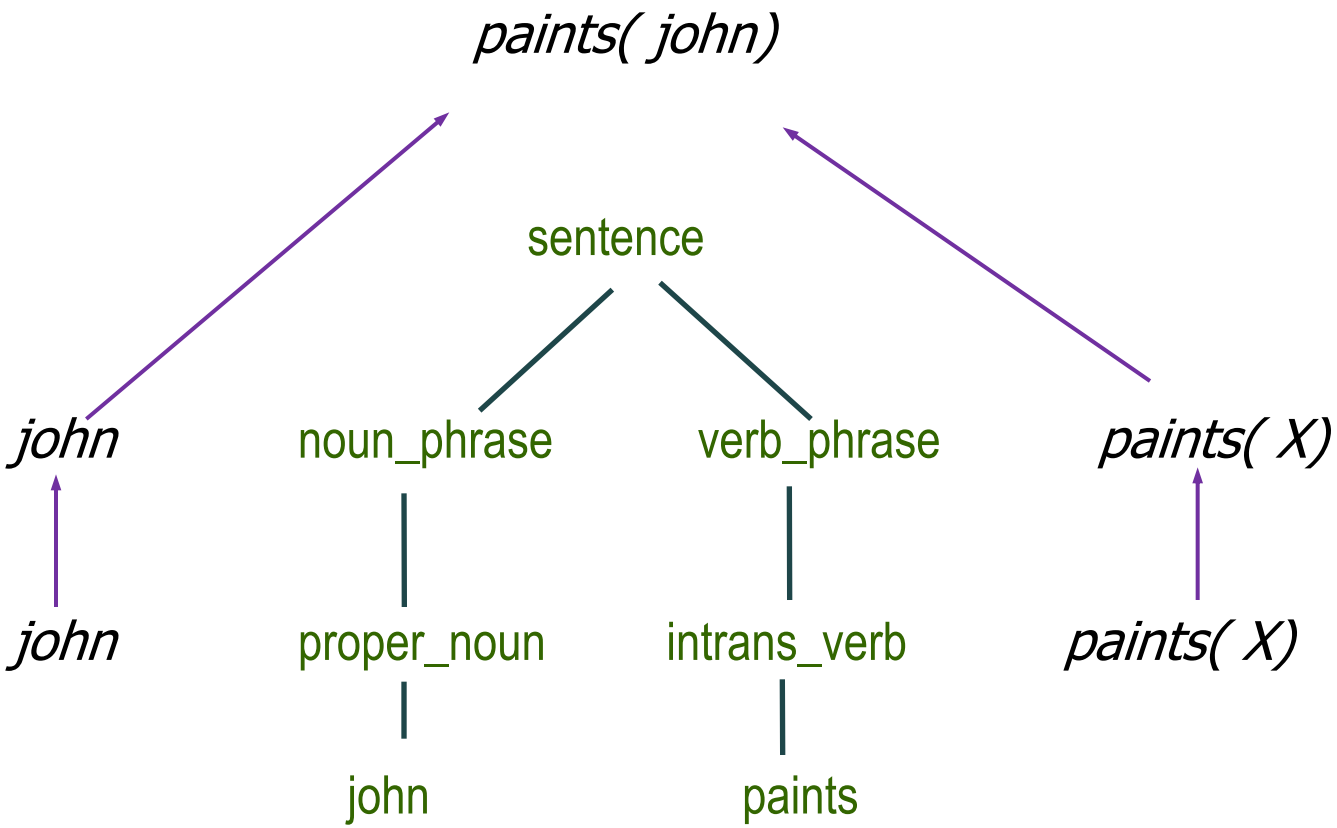
```
% "john" means "john"

proper_noun( john) --> [ john].

% "paints" means "paints( X)"

intrans_verb( paints( X)) --> [ paints].
```

SESTAVLJANJE POMENOV SAMOSTALNIŠKE FRAZE IN GLAGOLSKE FRAZE

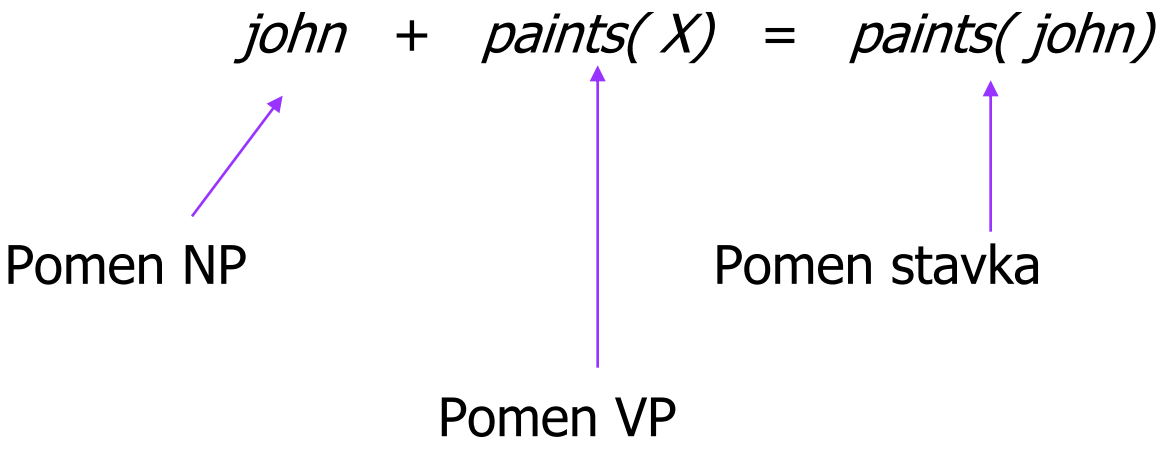
“John paints”



SESTAVLJANJE POMENOV SAMOSTALNIŠKE FRAZE IN GLAGOLSKE FRAZE



SESTAVLJANJE POMENOV FRAZ



SESTAVLJANJE POMENOV FRAZ

```
noun_phrase( NP)  -->  proper_noun( NP).

verb_phrase( VP)  -->  intrans_verb( VP).

sentence( S)  --->
    noun_phrase( NP),  verb_phrase( VP),
    { compose( NP, VP, S) } .
```

SESTAVLJANJE POMENOV NP IN VP

```
compose( NP, VP, VP)  :-
  arg( 1, VP, NP).              % 1st argument in VP is NP
```

```
?-  compose( john, paints( X), S).
S  =  paints( john)
```

BOLJ ELEGANTNO

DEFINIRAJ POMEN GLAGOLA TAKO,
DA OSEBEK (AGENT) POSTANE DODATNI ARGUMENT

```
intrans_verb( Agent, paints( Agent))  -->  [ paints].

verb_phrase( Agent, VP)  -->  intrans_verb( Agent, VP).

sentence( VP)  --->
    noun_phrase( Agent),  verb_phrase( Agent, VP).
```

S tem naredimo “predale“ vidne od zunaj

paints(Agent)

Predal, ki ga zapolnimo vsebino iz konteksta

POMEN TRANZITIVNIH GLAGOLOV

- “John likes Mary” pomeni *likes(john, mary)*
- “likes” pomeni *likes(Somebody, Something)*

```
trans_verb( Somebody, Something, likes(Somebody,Something)) -->
      [ likes].

verb_phrase( Somebody, VP) -->
    trans_verb( Somebody, Something, VP),
    noun_phrase( Something).
```

POMEN „DOLOČILNIKOV“
“A” IN “EVERY”

- “A man paints” pomeni:

  exists(X, man(X) and paints(X))

- Določilnik “a” diktira pomen celega stavka!

- “a man” pomeni:

  exists(X, man(X) and Assertion)

Trditev o X

KAJ POMENI “A”?

- “a” pomeni:

  There is some X such that:

  X has some property and

  some assertion holds about X.

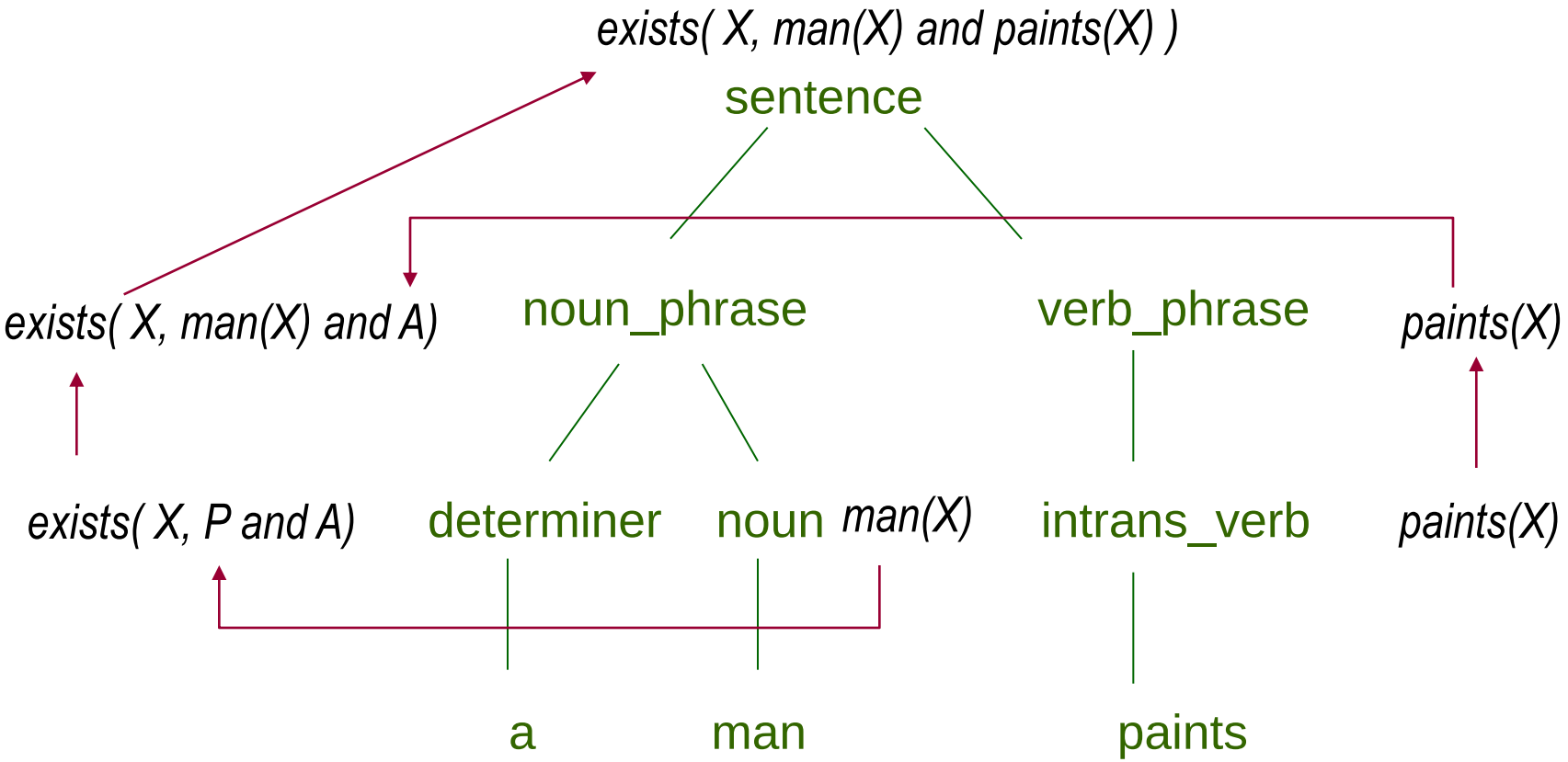
  exists(X, Property and Assertion)

- Naredimo X, Property in Assertion vidne:

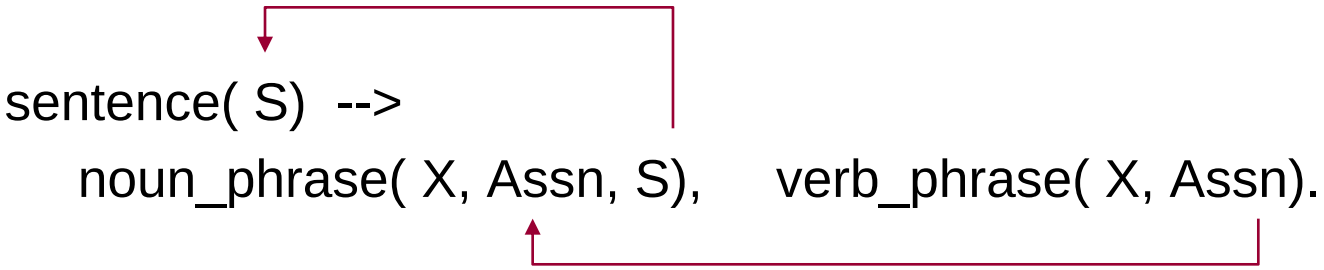
  determiner(X, Property, Assn, exists(X, Property and Assn)) -->

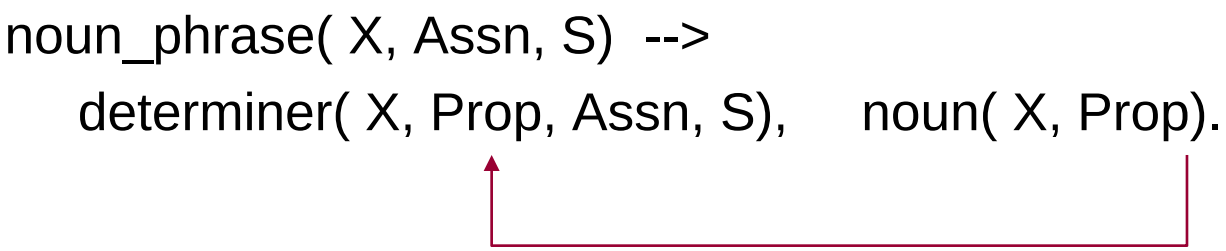
  [a].

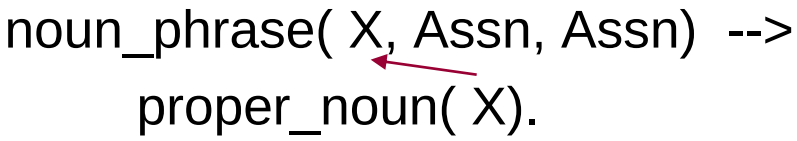
INTEGRACIJA POMENOV DOLOČILNIKOV IN DRUGIH FRAZ



INTEGRACIJA POMENOV DOLOČILNIKOV IN DRUGIH FRAZ







INTEGRACIJA POMENOV DOLOČILNIKOV IN DRUGIH FRAZ

```
verb_phrase( X, Assn)  -->
  intrans_verb( X, Assn).

determiner( X, Prop, Assn, exists( X, Prop and Assn)) --> [a].

noun( X, man( X))  -->  [ man].

intrans_verb( X, paints(X))  -->  [paints].
```

DOLOČILNIK “EVERY”

“Every woman dances” means:

all(X, woman(X) ==> dances(X))

determiner(X, Prop, Assn, all(X, Prop ==> Assn))
--> [every].

RELATIVNI STAVKI

“Every man that paints admires Monet”

SINTAKSA:

```
noun_phrase  -->  determiner,  noun,  rel_clause.

rel_clause  -->  [ that],  verb_phrase.

rel_clause  -->  [ ].    % Empty relative clause
```

RELATIVNI STAVKI

“Every man that paints admires Monet”

POMEN:

all(X, man(X) and paints(X) ==> admires(X, monet))

NALOGA

Definiraj v logiki pomen naslednjih stavkov:

(a) Every student of Newton did an experiment that involved gravity.

(b) Every dog chases a cat that hates the dog.

(c) Every dog chases a cat that hates a dog.

NALOGA, NAD.

(d) Every Italian cooks pasta.

(e) Every Italian in Peartree Village owns a restaurant that overlooks a bridge.

(f) A lady from the Information Office helps all customers that book a tour.

RELATIVNI STAVKI

POMEN, SPLOŠNA OBLIKA

all(X, Prop1 and Prop2 ==> Assertion)

noun — verb phrase of rel. clause — verb phrase of sentence

RELATIVNI STAVKI: SINTAKSA IN POMEN

```
rel_clause( X, Prop1, Prop1  and  Prop2)   -->
      [ that ],  verb_phrase( Prop2).

noun_phrase( X, Assn, S)   -->
      determiner( X, Prop12, Assn, S),  noun( X, Prop1),
      rel_clause( X, Prop1, Prop12).

rel_clause( X, Prop1, Prop1)   -->   [ ].
```

KOMPLETNA GRAMATIKA

```
%  Meaning of subset of natural language
:-  op( 100, xfy, and).
:-  op( 150, xfy, =>).

sentence( S)  -->
  noun_phrase( X, P, S), verb_phrase( X, P).

noun_phrase( X, P, S)  -->
  determiner( X, P12, P, S), noun( X, P1), rel_clause( X, P1, P12).

noun_phrase( X, P, P)  -->
  proper_noun( X).

verb_phrase( X, P)  -->
  trans_verb( X, Y, P1), noun_phrase( Y, P1, P).

verb_phrase( X, P)  -->
  intrans_verb( X, P).
```

KOMPLETNA GRAMATIKA, NAD.

```
rel_clause( X, P1, P1 and P2)  -->
  [that], verb_phrase( X, P2).

rel_clause( X, P1, P1)  --> [].

determiner( X, P1, P, all( X, P1 => P))  --> [every].

determiner( X, P1, P, exists( X, P1 and P)) --> [a].

noun( X, man(X))  -->  [man].
noun( X, woman(X))  -->  [woman].
proper_noun( john)  -->  [john].
proper_noun( annie)  -->  [annie].
proper_noun( monet)  -->  [monet].
trans_verb( X, Y, likes( X, Y))  -->  [ likes].
trans_verb( X, Y, admires( X, Y))  -->  [admires].
intrans_verb( X, paints(X))  -->  [paints].
```

PRIMERI STAVKOV,
KI JIH RAZUME TA GRAMATIKA

“Every man that paints admires Monet”

“Annie admires every man that paints”

“Every woman that admires a man that paints likes Monet”

Gramatika za ta stavek zgradi naslednji pomen:

all(X, woman(X) and exists(Y, (man(Y) and paints (Y))
and admires(X, Y)) ==> likes(X, monet))

TESTNI STAVKI

```
% Some tests

test1( M)  :-
  sentence( M, [john,paints],[]).

test2( M)  :-
  sentence( M, [a, man, paints], []).

test3( M)  :-
  sentence( M, [every,man,that,paints,admires,monet],[]).

test4( M)  :-
  sentence( M, [annie,admires,every,man,that,paints],[]).

test5( M)  :-
  sentence( M, [every,woman,that,admires,a,man,that,paints,likes,monet],[]).
```

NALOGA

Naslednje pravilo skuša definirati pomen določilnika „every“:

determiner(X, Prop, Assn, all(X, Prop and Assn)) --> [every].

(a) Ali je pravilo pravilno? Če ni, predlagaj popravek.

(b) Kakšno vlogo imata argumenta Prop in Assn? Katere stavčne fraze določajo vrednost Prop in Assn?

UPORABA V ODGOVARJANJU NA VPRAŠANJA

Pomeni v logiki se lahko izrazijo tudi v prologu:

```
admires( X, monet)  :-
    man( X),  paints( X).

admires( annie, X)  :-
     man( X),  paints( X).

likes( X, monet)  :-
    woman( X), man( Y), paints( Y), admires( X, Y).
```

“Does Annie admire anybody who admires Monet?”

```
?-  admires( annie, X),  admires( X, monet).
```

GENERIRANJE STAVKOV

- Naša gramatika lahko tudi generira stavke v z dani pomenom:

?- sentence(likes(john,annie), S, []).

S = [john, likes, annie]

?- sentence(exists(X, man(X) and paints(X)), S, []).

S = [a, man, paints]

PREVAJANJE MED JEZIKI Z DCG

- Prevod iz jezika 1 v jezik 2:

```
?-  ...,
    sentence_lang1( Meaning, Sent1, []),
    sentence_lang2( Meaning, Sent2, []).
```

- Lahko uporabimo tudi za prevajanje programskih jezikov

```
?- ...,
   program_source( M, SourceProg, []),    % Parse
   program_target( M, TargetProg, []).      % Generate
```

KOMPLEKSNOST NARAVNEGA JEZIKA

An example that illustrates how hard it is to handle NL perfectly. The meaning also depends on the stress – which word in a sentence is stressed. Here is an example in which EACH word can be stressed, and each variant has different meaning:

I never said she stole my money.

Possibilities are:

I never said she stole my money.
I never said she stole my money.
I never said she stole my money.
I never said she stole my money.
I never said she stole my money.
I never said she stole my money.
I never said she stole my money.